Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA

Alexander Bertrand and Marc Moonen

KU Leuven, Dept. of Electrical Engineering ESAT
Stadius Center for Dynamical Systems, Signal Processing and Data Analytics
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium

E-mail: alexander.bertrand@esat.kuleuven.be
marc.moonen@esat.kuleuven.be

Phone: +32 16 321899, Fax: +32 16 321970

***Abstract*—We describe a distributed adaptive algorithm to estimate the eigenvectors corresponding to the Q largest or smallest eigenvalues of the network-wide sensor signal covariance matrix in a wireless sensor network. The proposed algorithm recursively updates the eigenvector estimates without explicitly constructing the full covariance matrix that defines them, i.e., without centralizing all the raw sensor signal observations. By only sharing fused Q-dimensional observations, each node obtains estimates of (a) the node-specific entries of the Q covariance matrix eigenvectors, and (b) Q-dimensional projections of the full set of sensor signal observations onto the Q eigenvectors. We also explain how the latter can be used for, e.g., compression and reconstruction of the sensor signal observations based on principal component analysis (PCA), in which each node acts as a data sink. We describe a version of the algorithm for fully-connected networks, as well as for partially-connected networks. In the latter case, we assume that the network has been pruned to a tree topology to avoid cycles in the network. We provide convergence proofs, as well as numerical simulations to demonstrate the convergence and optimality of the algorithm.**

EDICS: SAM-MCHA Multichannel processing, SEN Signal Processing for Sensor Networks

***Index Terms*—Wireless sensor networks (WSNs), distributed estimation, principal component analysis**

The work of A. Bertrand was supported by a Postdoctoral Fellowship of the Research Foundation - Flanders (FWO). This work was carried out at the ESAT Laboratory of KU Leuven, in the frame of KU Leuven Research Council CoE EF/05/006 'Optimization in Engineering' (OPTEC) and PFV/10/002 (OPTEC), Concerted Research Action GOA-MaNet, the Belgian Programme on Interuniversity Attraction Poles initiated by the Belgian Federal Science Policy Office IUAP P7/23 (BESTCOM, 2012-2017) and IUAP P7/19 (DYSCO, 2012-2017), Research Project FWO nr. G.0763.12 'Wireless acoustic sensor networks for extended auditory communication' and project HANDiCAMS. The project HANDiCAMS acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 323944. The scientific responsibility is assumed by its authors. A conference precursor of this manuscript has been published as [1].

I. Introduction

A. Context and contribution

The eigenvectors of a signal covariance matrix play an important role in many algorithms and applications, e.g., in principal component analysis (PCA) [2], [3], the Karhunen-Loeve transform (KLT) [4], steering vector or direction-of-arrival estimation [5], [6], total least squares (TLS) estimation [7], subspace estimation, etc. In this paper, we address the estimation of the eigenvectors of the network-wide sensor signal covariance matrix in a wireless sensor network (WSN). Assume node k collects observations of a node-specific stochastic vector $\mathbf{y}_k$ and let $\mathbf{y}$ be the vector in which all $\mathbf{y}_k$'s are stacked. Our goal is then to adaptively estimate the Q eigenvectors corresponding to the Q largest or smallest eigenvalues of the network-wide covariance matrix defined by $\mathbf{y}$. In principle, this would require each node to transmit its raw sensor signal observations to a central node or fusion center (FC), where the network-wide covariance matrix can be constructed, after which an eigenvalue decomposition (EVD) can be performed. However, centralizing all these raw observations may require too much communication bandwidth, in particular if observations are collected at a high sampling rate, as in audio or video applications. Furthermore, if $\mathbf{y}$ has a large dimension, the computation of the EVD of the network-wide covariance matrix may require a significant amount of computational power at the FC since the computational complexity of the EVD scales cubically with the matrix dimension.

To reduce the communication and computation cost, we propose a distributed adaptive algorithm to estimate Q eigenvectors without explicitly constructing the network-wide covariance matrix that actually defines them, i.e., without the need to centralize the sensor signal observations in an FC. Instead of transmitting all its raw sensor signal observations, each node only transmits fused/compressed Q-dimensional observations, while estimating the node-specific entries of the eigenvector, corresponding to the part of $\mathbf{y}$ that is observed at the node. We refer to the algorithm as the distribute

adaptive covariance matrix eigenvector estimation (DACMEE) algorithm.

The DACMEE algorithm also provides each node with the Q-dimensional projections of the full set of sensor signal observations onto the Q estimated eigenvectors. This allows each node to compute a PCA- or KLT-based approximation of the observations of the full network-wide vector $\mathbf{y}$.

We will describe two versions of the DACMEE algorithm, i.e., a version for fully-connected networks in which a signal broadcast by any node can be collected by every other node, as well as a version for partially-connected networks in which a node can only communicate with a subset of the other nodes. We then assume that the partially-connected network is pruned to a tree topology. This guarantees that there are no cycles in the network graph, since these harm the algorithm dynamics.

B. Relation to prior work

Two different cases have been considered in the literature where either (a) the nodes collect observations of the full vector $\mathbf{y}$, or (b) each node collects observations of a node-specific subset of the entries of $\mathbf{y}$ (as it is the case in this paper). Let $\mathbf{Y}$ denote an $M \times N$ observation matrix containing N observations of an M-dimensional stochastic vector $\mathbf{y}$, then (a) corresponds to the case where the *columns* of $\mathbf{Y}$ are distributed over the different nodes, whereas in case (b), the *rows* of $\mathbf{Y}$ are distributed over the nodes. The techniques to construct the corresponding covariance matrix and/or estimate its eigenvectors are very different for the two cases. It is noted that a similar distinction exists in the literature in the context of distributed least-squares estimation, see, e.g., [8], [9], where each node collects observations of the full $\mathbf{y}$ to estimate a common parameter vector, versus [10], [11], where each node only observes a node-specific subset of the entries of $\mathbf{y}$ to compute an estimator that relies on the full network-wide covariance matrix.

Case (a) is addressed in [12]–[14] for ad hoc topologies and in [15] for a fully-connected topology. In [12], the network-wide covariance matrix is first estimated by means of a consensus averaging (CA) algorithm that exchanges $M \times M$ matrices in each iteration, after which each node performs a local EVD. If only a subset of the eigenvectors[1] is needed, one can use distributed optimization techniques in which only M-dimensional vectors are exchanged between nodes [13], [14]. In [15], a distributed QR decomposition is performed in a fully-connected network, followed by an EVD.

Case (b) is actually more challenging, as it requires to estimate the cross-correlation between sensor signals of different nodes. This requires the exchange of (compressed) sensor signal observations, resulting in a higher communication cost compared to case (a), in particular for applications with a high sampling rate. Case (b) is tackled in [6], [16], [17] (only for the case of principal eigenvectors) for networks with an ad-hoc topology. These algorithms rely on Oja's learning rule in combination with nested CA iterations, hence operating at two time scales. The inner loop performs many CA iterations with a full reset for each outer loop iteration. Since the outer loop runs with the same rate as the sampling rate of $\mathbf{y}$, and since each iteration of the inner loop also requires data exchange, each node actually transmits more data than actually collected by its sensors. Furthermore, since the convergence time of the inner CA loop increases with the network size, the per-node communication cost also grows with the network size.

The algorithm proposed in this paper only works in networks with a fully connected or a tree topology, but it does not require nested loops, and its per-node communication cost is independent of the network size. The algorithm does not explicitly rely on Oja's stochastic learning rule (although this can also be included), but it explicitly computes compressed sensor signal covariance matrices at each node. The latter allows to, e.g., remove the effect of spatially correlated noise by subtracting a known or estimated noise covariance matrix from the local covariance matrices. The algorithm is also able to estimate the eigenvectors corresponding to the smallest eigenvalues (e.g., for TLS estimation), which is not possible in [6], [16], [17].

Finally, it is noted that there exists other related work in the context of (b) (see, e.g., [3], [4]), which however requires prior knowledge of the network-wide covariance matrix. In our case, the network-wide covariance matrix is assumed to be unknown (and possibly even time-varying).

C. Paper outline

The outline of the paper is as follows. Section II gives the problem statement as well as an application example in the context of distributed PCA in a WSN. Section III describes the DACMEE algorithm for fully-connected networks and provides convergence and optimality proofs. Section IV extends these results towards networks with a tree topology. The theoretical results are validated by means of Monte-Carlo simulations in Section V. Conclusions are drawn in Section VI.

II. Preliminaries

A. Problem statement

Consider a WSN with a set of sensor nodes $\mathcal{K} = \{1, \ldots, K\}$ (we do not make any assumptions on the network topology at this point). Node k collects observations of an M_k-dimensional stochastic vector $\mathbf{y}_k$, which is assumed to be (short-term[2]) stationary and ergodic. We assume that $\mathbf{y}_k$ is complex-valued to allow for a frequency-domain description, e.g., in the short-time Fourier transform (STFT) domain. We define the M-dimensional stochastic vector $\mathbf{y}$ as the stacked version of all $\mathbf{y}_k$'s, where $M = \sum_{k \in \mathcal{K}} M_k$. For the sake of an easy exposition, but without loss of generality (w.l.o.g.), we assume that $\mathbf{y}$ is zero-mean, which may require a mean subtraction pre-processing step. The covariance matrix of $\mathbf{y}$ is then defined as

$$\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\} \tag{1}$$

[1]The algorithms in [13], [14] estimate the eigenvector corresponding to the smallest eigenvalue of the covariance matrix, but the algorithms are easily adapted to compute the principal eigenvectors.

[2]Since the algorithms envisaged in this paper are adaptive, short-term stationarity is sufficient.

where $E\{\cdot\}$ denotes the expected value operator, and the superscript H denotes the conjugate transpose operator. Let $\mathbf{Y}$ denote an $M \times N$ observation matrix containing N different observations of $\mathbf{y}$ in its columns. Then ergodicity of $\mathbf{y}$ implies that $\mathbf{R}_{yy}$ can be approximated by the sample covariance matrix, i.e.,

$$\mathbf{R}_{yy} \approx \frac{1}{N}\mathbf{Y}\mathbf{Y}^H \tag{2}$$

and equality holds in the case of an infinite observations window, i.e., $\mathbf{R}_{yy} = \lim_{N\to\infty} \frac{1}{N}\mathbf{Y}\mathbf{Y}^H$.

The eigenvalue decomposition (EVD) of $\mathbf{R}_{yy}$ is defined as

$$\mathbf{R}_{yy} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^H \tag{3}$$

where $\mathbf{\Sigma} = \text{diag}(\lambda_1, \ldots, \lambda_M)$ is a real diagonal matrix with the eigenvalues as its diagonal elements (sorted in decreasing order of magnitude), and where the unitary matrix $\mathbf{U}$ contains the corresponding (normalized) eigenvectors in its columns. Consider the following truncated versions of $\mathbf{U}$

$$\hat{\mathbf{X}} = \mathbf{U}\begin{bmatrix} \mathbf{I}_Q \\ \mathbf{O}_{(M-Q)\times Q} \end{bmatrix} \tag{4}$$

and

$$\check{\mathbf{X}} = \mathbf{U}\begin{bmatrix} \mathbf{O}_{(M-Q)\times Q} \\ \mathbf{I}_Q \end{bmatrix} \tag{5}$$

where $\mathbf{I}_Q$ denotes the $Q \times Q$ identity matrix and $\mathbf{O}_{(M-Q)\times Q}$ denotes the $(M-Q)\times Q$ all-zero matrix. $\hat{\mathbf{X}}$ is then a $M \times Q$ matrix with the Q principal eigenvectors of $\mathbf{R}_{yy}$ in its columns, i.e., the eigenvectors corresponding to the Q largest eigenvalues. Similarly, $\check{\mathbf{X}}$ contains the Q eigenvectors corresponding to the Q smallest eigenvalues.

The principal eigenvectors in $\hat{\mathbf{X}}$ can be used for compression in the context of PCA or KLT (we will briefly illustrate this in Subsection II-B) or for steering vector and/or direction-of-arrival estimation [5], [6]. It is noted that $\hat{\mathbf{X}}$ is also a solution of the following constrained optimization problem

$$\hat{\mathbf{X}} \in \arg\max_{\mathbf{X}} \text{Tr}\left\{\mathbf{X}^H\mathbf{R}_{yy}\mathbf{X}\right\} \tag{6}$$

$$\text{s.t. } \mathbf{X}^H\mathbf{X} = \mathbf{I}_Q \tag{7}$$

where $\text{Tr}\{.\}$ denotes the trace operator yielding the sum of the diagonal elements of the matrix in its argument. This formulation will be exploited in the derivation of the DACMEE algorithm in Section III.

In the sequel, we will only consider the distributed estimation of $\hat{\mathbf{X}}$. However, it should be noted that all results in this paper can be straightforwardly modified to also compute $\check{\mathbf{X}}$, which can be used for, e.g., total least squares estimation [7] or null-space estimation. This modification only involves replacing the $\max$ operator with a $\min$ operator in (6) and in similar expressions in the sequel.

The DACMEE algorithm is a distributed adaptive (time-recursive) algorithm where each node is responsible for updating a specific part of $\hat{\mathbf{X}}$, avoiding the centralized computation of the full covariance matrix $\mathbf{R}_{yy}$. Each node only transmits Q-dimensional compressed observations instead of the M_k-dimensional observations of $\mathbf{y}_k$. In the case of fully-connected networks, this significantly reduces the communication bandwidth and the per-node computational complexity if $Q \ll M_k$. In case of multi-hop networks, here assumed to be pruned to a tree topology[3], the DACMEE algorithm yields a significant reduction in bandwidth and computational complexity even for $Q \geq M_k$ (when compared to the relay case where the nodes relay all observations to a data sink).

For the sake of an easy exposition, but w.l.o.g., we make the pragmatic assumption throughout this paper that $\hat{\mathbf{X}}$ is unique up to a scaling ambiguity in its columns. This means that the $Q+1$ largest eigenvalues of $\mathbf{R}_{yy}$ are non-degenerate (they all have a different value). This allows to make compact claims without each time having to differentiate between the degenerate case and the non-degenerate case. The case where $\hat{\mathbf{X}}$ is not unique is briefly addressed in Appendix VII-D.

B. Application example: PCA-based distributed compression

If the entries in $\mathbf{y}$ are sufficiently correlated, the matrix $\hat{\mathbf{X}}$ can be used as a compression matrix to compress the M-dimensional observations in the columns of $\mathbf{Y}$ into Q-dimensional observations, i.e., $\hat{\mathbf{Z}} = \hat{\mathbf{X}}^H\mathbf{Y}$. The reconstruction can then be implemented by using $\hat{\mathbf{X}}$ as a decompression matrix, i.e., $\mathbf{Y} \approx \hat{\mathbf{X}}\hat{\mathbf{Z}}$. This is the main principle behind PCA [2], [3] and the KLT [4], and it can be shown that $\hat{\mathbf{X}}$ is the best linear compressor for $\mathbf{Y}$ in minimum mean squared error (MMSE) sense.

If $\hat{\mathbf{X}}$ is known, the coefficients of this matrix can be communicated to the nodes such that future observations can be compressed using $\hat{\mathbf{X}}$. To this end, we define the block partitioning

$$\hat{\mathbf{X}} = \begin{bmatrix} \hat{\mathbf{X}}_1 \\ \vdots \\ \hat{\mathbf{X}}_K \end{bmatrix} \tag{8}$$

where $\hat{\mathbf{X}}_k$ is the part of $\hat{\mathbf{X}}$ that is applied to $\mathbf{y}_k$ such that $\hat{\mathbf{X}}^H\mathbf{y} = \sum_{k\in\mathcal{K}} \hat{\mathbf{X}}_k^H\mathbf{y}_k$. Node k can then transmit observations of the Q-dimensional (compressed) stochastic variable $\mathbf{z}_k = \hat{\mathbf{X}}_k^H\mathbf{y}_k$ instead of observations of $\mathbf{y}_k$. This allows for a distributed compress-and-fuse framework, where the Q-dimensional observations of the different $\mathbf{z}_k$'s are fused on their way through the network by simple addition to finally obtain $\hat{\mathbf{z}} = \sum_{k\in\mathcal{K}} \mathbf{z}_k = \hat{\mathbf{X}}^H\mathbf{y}$ at the data sink. The DACMEE algorithm is designed such that $\hat{\mathbf{z}}$ is available to each node, i.e., each node acts as a data sink. The original observations can then be reconstructed as $\mathbf{y} \approx \hat{\mathbf{X}}\hat{\mathbf{z}}$.

III. The DACMEE algorithm in fully-connected networks

A. Algorithm derivation

The DACMEE algorithm is an iterative algorithm that updates the $M \times Q$ matrix $\mathbf{X}^i$, where i is the iteration index. In each iteration, N new sensor signal observations will be used to update $\mathbf{X}^i$ in a time-recursive fashion. We again define the block partitioning as in (8)

$$\mathbf{X}^i = \begin{bmatrix} \mathbf{X}_1^i \\ \vdots \\ \mathbf{X}_K^i \end{bmatrix} \tag{9}$$

[3]This will be motivated in Section IV.

where the entries of the block $\mathbf{X}_k^i$ will be updated by node k. The goal is to find the Q principal eigenvectors of $\mathbf{R}_{yy}$, i.e., to obtain that $\lim_{i\to\infty}\mathbf{X}_k^i = \hat{\mathbf{X}}_k$, $\forall\, k \in \mathcal{K}$, by letting nodes exchange compressed observations of their $\mathbf{y}_k$'s.

The DACMEE algorithm basically consists of an iterative procedure that gradually increases the value of the objective function

$$J(\mathbf{X}) = \text{Tr}\left\{\mathbf{X}^H\mathbf{R}_{yy}\mathbf{X}\right\} \tag{10}$$

under the orthogonality constraint $\mathbf{X}^H\mathbf{X} = \mathbf{I}_Q$. As mentioned in Section II-A (see (6)-(7)), $\hat{\mathbf{X}}$ is indeed a maximum of this optimization problem. The algorithm derivation starts from the following alternating optimization (AO) procedure[4]:

1) Set $i \leftarrow 0$, $q \leftarrow 1$, and choose $\mathbf{X}^0$ as a random $M \times Q$ matrix.
2) Choose $\mathbf{X}^{i+1}$ as a solution of:

$$\mathbf{X}^{i+1} \in \arg\max_{\mathbf{X}} J(\mathbf{X}) \tag{11}$$

$$\text{s.t.} \quad \cdot\ \mathbf{X}^H\mathbf{X} = \mathbf{I}_Q \tag{12}$$

$$\cdot\ \forall\, k \in \mathcal{K}\backslash\{q\}:\ \mathbf{X}_k \in \text{Range}\{\mathbf{X}_k^i\} \tag{13}$$

   where $\mathbf{X}_k$ is the k-th submatrix of $\mathbf{X}$ similarly defined as in (9), and where $\text{Range}\{\mathbf{X}_k^i\}$ denotes the subspace spanned by the columns of $\mathbf{X}_k^i$.
3) $i \leftarrow i+1$ and $q \leftarrow (q \mod K) + 1$.
4) Return to step 2.

Note that each iteration of this AO procedure increases the objective function $J\left(\mathbf{X}^{i+1}\right)$ in a monotonic fashion while adhering to the orthogonality constraint $\mathbf{X}^H\mathbf{X} = \mathbf{I}_Q$. Indeed, the constraint (13) changes in each iteration, allowing to update a particular submatrix of $\mathbf{X}$ freely (i.e., $\mathbf{X}_q$), while constraining the other submatrices to preserve their current column space. Despite the fact that this AO procedure is a centralized procedure requiring the full covariance matrix $\mathbf{R}_{yy}$, the particular form of the constraints (13) allows to execute it in a distributed fashion, which is explained next.

We define

$$\mathbf{B}_{<k}^i = \text{Blkdiag}\left(\mathbf{X}_1^i, \ldots, \mathbf{X}_{k-1}^i\right) \tag{14}$$

$$\mathbf{B}_{>k}^i = \text{Blkdiag}\left(\mathbf{X}_{k+1}^i, \ldots, \mathbf{X}_K^i\right) \tag{15}$$

where $\text{Blkdiag}\left(\cdot\right)$ is an operator that generates a block diagonal matrix[5] from the matrices in its argument. Using this, we define the matrix $\mathbf{C}_k^i$ as

$$\mathbf{C}_k^i = \left[\begin{array}{c|c|c} \mathbf{O} & \mathbf{B}_{<k}^i & \mathbf{O} \\ \hline \mathbf{I}_{M_k} & \mathbf{O} & \mathbf{O} \\ \hline \mathbf{O} & \mathbf{O} & \mathbf{B}_{>k}^i \end{array}\right] \tag{16}$$

where $\mathbf{O}$ is an all-zero matrix of appropriate dimension. Using this matrix, we can eliminate the constraint (13) by parameterizing $\mathbf{X}$ as $\mathbf{X} = \mathbf{C}_q^i\widetilde{\mathbf{X}}$, where $\widetilde{\mathbf{X}}$ becomes the new optimization variable (note that the unconstrained subblock $\mathbf{X}_q$ corresponds to the first M_q rows of $\widetilde{\mathbf{X}}$). Indeed, solving (11)-(13) is then equivalent to solving

$$\widetilde{\mathbf{X}} \in \arg\max_{\widetilde{\mathbf{X}}} \text{Tr}\left\{\widetilde{\mathbf{X}}^H\mathbf{C}_q^{i\,H}\mathbf{R}_{yy}\mathbf{C}_q^i\widetilde{\mathbf{X}}\right\} \tag{17}$$

$$\text{s.t.} \quad \widetilde{\mathbf{X}}^H\mathbf{C}_q^{i\,H}\mathbf{C}_q^i\widetilde{\mathbf{X}} = \mathbf{I}_Q \tag{18}$$

and setting $\mathbf{X}^{i+1} = \mathbf{C}_q^i\widetilde{\mathbf{X}}$.

It is noted that $\mathbf{C}_q^{i\,H}\mathbf{R}_{yy}\mathbf{C}_q^i$ can be interpreted as the covariance matrix of a compressed version of the sensor signals in $\mathbf{y}$, where $\mathbf{C}_q^i$ is used as a compression matrix. Indeed, define

$$\widetilde{\mathbf{y}}_k^i = \mathbf{C}_k^{i\,H}\mathbf{y}\ , \tag{19}$$

then

$$\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i \triangleq E\{\widetilde{\mathbf{y}}_k^i\widetilde{\mathbf{y}}_k^{i\,H}\} = \mathbf{C}_k^{i\,H}\mathbf{R}_{yy}\mathbf{C}_k^i\ . \tag{20}$$

This means that we can rewrite (17)-(18) as

$$\widetilde{\mathbf{X}} \in \arg\max_{\widetilde{\mathbf{X}}} \text{Tr}\left\{\widetilde{\mathbf{X}}^H\mathbf{R}_{\tilde{y}_q\tilde{y}_q}^i\widetilde{\mathbf{X}}\right\} \tag{21}$$

$$\text{s.t.} \quad \widetilde{\mathbf{X}}^H\mathbf{C}_q^{i\,H}\mathbf{C}_q^i\widetilde{\mathbf{X}} = \mathbf{I}_Q\ . \tag{22}$$

The DACMEE algorithm will exploit the compression of $\mathbf{y}$ based on (19), by letting each node $k \in \mathcal{K}$ use its local $\mathbf{X}_k^i$ as a compression matrix to compress the observations of the M_k-channel signal $\mathbf{y}_k$ into a Q-channel signal[6]

$$\mathbf{z}_k^i = \mathbf{X}_k^{i\,H}\mathbf{y}_k \tag{23}$$

and we define $\mathbf{z}^i = [\mathbf{z}_1^{i\,T}\ \ldots\ \mathbf{z}_K^{i\,T}]^T$ as the stacked version of all the $\mathbf{z}_k^i$'s. In between iterations i and $i+1$ of the DACMEE algorithm, each node $k \in \mathcal{K}$ will transmit new observations of the compressed signal $\mathbf{z}_k^i$ to the other nodes. Assuming a fully-connected network, then each node k collects observations of

$$\widetilde{\mathbf{y}}_k^i = \left[\begin{array}{c} \mathbf{y}_k \\ \mathbf{z}_{-k}^i \end{array}\right] \tag{24}$$

where $\mathbf{z}_{-k}^i$ is equal to $\mathbf{z}^i$ with $\mathbf{z}_k^i$ removed, i.e., $\mathbf{z}_{-k}^i = [\mathbf{z}_1^{i\,T}\ \ldots\ \mathbf{z}_{k-1}^{i\,T}\ \mathbf{z}_{k+1}^{i\,T}\ \ldots\ \mathbf{z}_K^{i\,T}]^T$. Note that the definition of $\widetilde{\mathbf{y}}_k^i$ in (24) is equivalent to (19). Assuming $\mathbf{C}_q^i$ is known to node q, then (21)-(22) can be solved locally by node q, since node q has access to observations of $\widetilde{\mathbf{y}}_q^i$. Indeed, similarly to (2), $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i$ can be estimated at node k, $\forall\, k \in \mathcal{K}$ as

$$\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i \approx \frac{1}{N}\widetilde{\mathbf{Y}}_k^i\widetilde{\mathbf{Y}}_k^{i\,H} \tag{25}$$

where $\widetilde{\mathbf{Y}}_k^i$ is an $(M_k + (K-1)Q) \times N$ matrix, containing N observations of $\widetilde{\mathbf{y}}_k^i$ in its columns.

As a final step in the algorithm derivation, we will transform (21)-(22) into an eigenvalue problem, which is then solved locally at node q by means of an EVD. To this end, we define the $Q \times Q$ matrix

$$\mathbf{D}_k^i = \mathbf{X}_k^{i\,H}\mathbf{X}_k^i \tag{26}$$

and its $Q \times Q$ square root factor

$$\mathbf{L}_k^i = \left(\mathbf{D}_k^i\right)^{\frac{1}{2}} \tag{27}$$

[4]The algorithm may stop when a certain stopping criterion is satisfied or it may run continuously, e.g., in an adaptive context where $\mathbf{R}_{yy}$ (slowly) changes across the iterations.

[5]We use the operator $\text{Blkdiag}\left(\cdot\right)$ with a slight abuse of notation, since a block-diagonal matrix in principle has square matrices as main diagonal blocks, which is not the case in (14)-(15).

[6]For the sake of an easy exposition, we assume that $Q < M_k$, $\forall\, k \in \mathcal{K}$. If there exists a k for which $Q \geq M_k$, node k should transmit uncompressed observations of $\mathbf{y}_k$ to one other node q, which can then add these to $\mathbf{y}_q$ as additional sensor signals.

which can be computed by means of, e.g., a Cholesky factorization [18] or an EVD (note that $\mathbf{L}_k^i$ is not unique as it depends on the factorization method that is used, but this choice has no influence on the algorithm). Finally, we define the block-diagonal matrix

$$\mathbf{\Lambda}_k^i = \text{Blkdiag}\left(\mathbf{I}_{M_k}, \mathbf{L}_1^i, \ldots, \mathbf{L}_{k-1}^i, \mathbf{L}_{k+1}^i, \ldots, \mathbf{L}_K^i\right) \quad (28)$$

and its inverse

$$\mathbf{V}_k^i = \left(\mathbf{\Lambda}_k^i\right)^{-1} . \quad (29)$$

Using the substitutions $\overline{\mathbf{X}} = \mathbf{\Lambda}_q^i \widetilde{\mathbf{X}}$ and $\overline{\mathbf{R}}_q^i = \mathbf{V}_q^{i\,H} \mathbf{R}_{\tilde{y}_q \tilde{y}_q}^i \mathbf{V}_q^i$, and using the fact that

$$\mathbf{\Lambda}_k^{i\,H} \mathbf{\Lambda}_k^i = \mathbf{C}_k^{i\,H} \mathbf{C}_k^i , \quad (30)$$

we find that (21)-(22) can be equivalently solved as

$$\overline{\mathbf{X}} \in \underset{\overline{\mathbf{X}}}{\arg\max}\, \text{Tr}\left\{\overline{\mathbf{X}}^H \overline{\mathbf{R}}_q^i \overline{\mathbf{X}}\right\} \quad (31)$$

$$\text{s.t.} \quad \overline{\mathbf{X}}^H \overline{\mathbf{X}} = \mathbf{I}_Q \quad (32)$$

and setting $\widetilde{\mathbf{X}} = \mathbf{V}_q^i \overline{\mathbf{X}}$. Note that (31)-(32) has the same form as (6)-(7), and hence it can be solved by computing an EVD of $\overline{\mathbf{R}}_q^i$ (note that the dimension of this EVD is much smaller than the network-wide EVD in (3)). Since $\mathbf{R}_{\tilde{y}_q \tilde{y}_q}^i$ can be estimated by node q based on (25), node q can compute $\overline{\mathbf{R}}_q^i$ and its EVD (assuming $\mathbf{V}_q^i$ is known, which will require exchange of the $\mathbf{L}_k^i$'s between the nodes).

Finally, we can conclude that the optimization step (11)-(13) of the original AO can be solved by setting $\mathbf{X}^{i+1} = \mathbf{C}_k^i \mathbf{V}_q^i \overline{\mathbf{X}}$, where the Q columns of $\overline{\mathbf{X}}$ contain the Q principal eigenvectors of $\overline{\mathbf{R}}_q^i$, which can be computed at node q. This allows to compute the AO in a distributed fashion by sequentially solving local EVD problems at the different nodes.

The resulting DACMEE algorithm for fully-connected networks is described in detail in Table I, and schematically illustrated in Fig. 1 (the shaded part is only active when the illustrated node k is the updating node at iteration i). Dashed lines indicate an exchange of parameters (at a slow rate), whereas full lines indicate exchange of signal observations (at a high rate). It is noted that the DACMEE algorithm as described in Table I is assumed to operate in an adaptive (time-recursive) context, where the different iterations are spread out over time, rather than retransmitting the same block of observations multiple times between nodes (which would reduce the communication efficiency). This can be seen in the time indices of $\mathbf{z}_k$ and $\mathbf{y}_k$ in step 3, which are indeed shifted with the iteration index i, hence each iteration happens over a new block of N observations.

In case the DACMEE algorithm is applied for PCA-based compression, the PCA-based compressed samples $\hat{\mathbf{X}}^H \mathbf{y}[iN+j]$ (with $j = 1, \ldots, N$) can be computed at each node as (see Subsection II-B)

$$\hat{\mathbf{z}}[iN+j] \triangleq \hat{\mathbf{X}}^H \mathbf{y}[iN+j] \approx \sum_{k \in \mathcal{K}} \mathbf{z}_k^i[iN+j] \quad (33)$$

where the approximation will improve over time when $i \to \infty$ (see also Subsection III-B). Each node can also reconstruct the observations of the network-wide vector $\mathbf{y}$ using $\mathbf{y}[iN+j] \approx \hat{\mathbf{X}}\hat{\mathbf{z}}[iN+j] \approx \mathbf{X}^i \sum_{k \in \mathcal{K}} \mathbf{z}_k^i[iN+j]$. This decompression requires some minor additional data exchange such that each node has access to the complete $\mathbf{X}^i$.

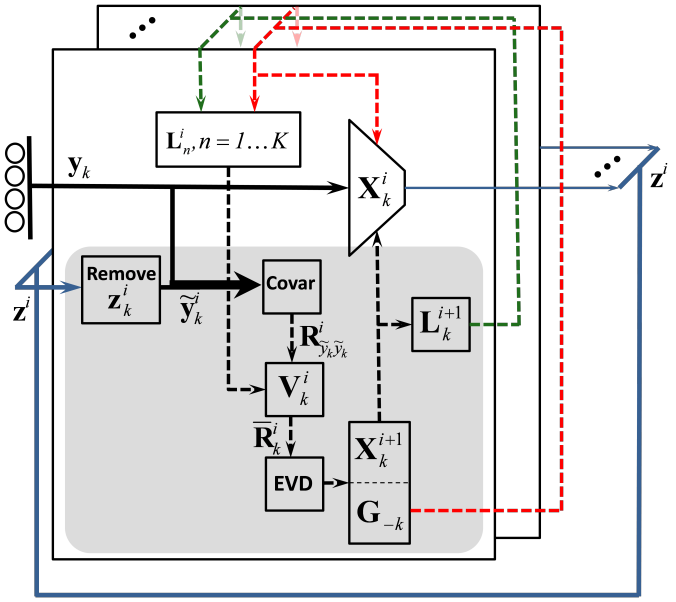

Fig. 1. Schematic illustration of the data flow and processing at node $k \in \mathcal{K}$ in the DACMEE algorithm in a fully connected network. The shaded part is only active when $k = q$, i.e., if node k is the updating node in step 4 of the DACMEE algorithm. Dashed lines indicate an exchange of parameters (at a slow rate), whereas full lines indicate exchange of signal observations (at a high rate).

Remark I) Trade-off between adaptation speed, estimation accuracy, and communication cost: It is noted that the number of observations N that are collected and broadcast in between the iterations (step 3) must be chosen such that a sufficiently accurate estimate of $\mathbf{R}_{\tilde{y}_k \tilde{y}_k}^i$ can be computed in step 4. This results in a trade-off between estimation accuracy and adaptation speed, as a larger N results in less frequent updates (per second) in the DACMEE algorithm (note that each iteration happens over a new block of N observations).

Furthermore, another trade-off can be introduced when computing multiple iterations of the DACMEE algorithm on the same block of N observations. Indeed, this would improve the adaptation speed and accuracy, but at the price of an increased communication cost.

Remark II) Additional communication of parameters: The last part of step 4 of the DACMEE algorithm in Table I involves an extra broadcasting of the $\mathbf{G}_{-q}$ matrix (defined in (35)) and $\mathbf{L}_q^{i+1}$. However, since $\mathbf{X}^i$ is only updated once for every block of N sensor signal observations, this extra broadcasting is negligible compared to the broadcasting of the KN (compressed) observations of the $\mathbf{z}_k^i$'s (step 3). This difference in communication rate is visualized in Fig. 1 by distinguishing between dashed lines (parameter exchange at a slow rate) and full lines (exchange of signal observations at a high rate). The same argument also holds if the full matrix $\mathbf{X}^i$ would be broadcast (e.g., for the purpose of decompression as explained in Subsection II-B).

TABLE I
DESCRIPTION OF THE DACMEE ALGORITHM IN A FULLY-CONNECTED NETWORK.

DACMEE algorithm in a fully-connected network

1) Set $i \leftarrow 0$, $q \leftarrow 1$, and initialize all $\mathbf{X}_k^0$, $\forall\, k \in \mathcal{K}$, with random entries.
2) Each node $k \in \mathcal{K}$ computes $\mathbf{D}_k^i = \mathbf{X}_k^{i\,H}\mathbf{X}_k^i$ and its square root factorization $\mathbf{L}_k^i = (\mathbf{D}_k^i)^{\frac{1}{2}}$, and broadcasts the $Q \times Q$ matrix $\mathbf{L}_k^i$.
3) Each node $k \in \mathcal{K}$ broadcasts N new compressed sensor signal observations $\mathbf{z}_k^i[iN+j] = \mathbf{X}_k^{i\,H}\,\mathbf{y}_k[iN+j]$ (where $j = 1 \dots N$).
4) At node q:
   - Compute $\mathbf{R}^i_{\tilde{y}_q\tilde{y}_q}$ based on (25).
   - Construct the block-diagonal matrix $\mathbf{\Lambda}_q^i$ as defined in (28) and compute the inverse of each diagonal block to construct the block-diagonal matrix $\mathbf{V}_q^i = \left(\mathbf{\Lambda}_q^i\right)^{-1}$.
   - Compute the Q principal eigenvectors of the matrix
   $$\overline{\mathbf{R}}_q^i = \mathbf{V}_q^{i\,H}\mathbf{R}^i_{\tilde{y}_q\tilde{y}_q}\mathbf{V}_q^i \tag{34}$$
   and construct $\overline{\mathbf{X}}_q^{i+1}$ with the eigenvectors in the columns, sorted in decreasing order of eigenvalue magnitudes.
   - Set
   $$\begin{bmatrix} \mathbf{X}_q^{i+1} \\ \mathbf{G}_{-q} \end{bmatrix} = \mathbf{V}_q^i\overline{\mathbf{X}}_q^{i+1} \tag{35}$$
   $$\mathbf{D}_q^{i+1} = \mathbf{X}_q^{i+1\,H}\mathbf{X}_q^{i+1}\ . \tag{36}$$
   - Compute the square root factorization $\mathbf{D}_q^{i+1} = \mathbf{L}_q^{i+1\,H}\mathbf{L}_q^{i+1}$.
   - Broadcast $\mathbf{L}_q^{i+1}$ and the matrix $\mathbf{G}_{-q}$.
5) Define the partitioning $\mathbf{G}_{-q} = \left[\mathbf{G}_1^T\ \dots\ \mathbf{G}_{q-1}^T\,\mathbf{G}_{q+1}^T\ \dots\ \mathbf{G}_K^T\right]^T$ where each $\mathbf{G}_k$ is a $Q \times Q$ matrix. Each node $k \in \mathcal{K}\backslash\{q\}$ updates
   $$\mathbf{L}_k^{i+1} = \mathbf{L}_k^i\mathbf{G}_k \tag{37}$$
   $$\mathbf{X}_k^{i+1} = \mathbf{X}_k^i\mathbf{G}_k\ . \tag{38}$$
6) $i \leftarrow i+1$ and $q \leftarrow (q \mod K) + 1$.
7) Return to step 3.

Note 1: The initial $\mathbf{X}^0$ does not have to satisfy $\mathbf{X}^{0\,H}\mathbf{X}^0 = \mathbf{I}_Q$, since the first iteration of the DACMEE algorithm will correct this.

Note 2: The implementation of the EVD of $\overline{\mathbf{R}}_q^i$ in step 4 is not specified. Rather than performing a cold reset of this EVD in each iteration, one can increase the 'time-recursiveness' of the algorithm by using time-recursive subspace estimation techniques instead, such as power iterations or Oja's learning rule [17], [19] (in each iteration initialized with the previous estimate $\left[\mathbf{X}_q^{i\,T}\,\mathbf{I}_Q\ \dots\ \mathbf{I}_Q\right]^T$).

Note 3: With a minor modification, the above algorithm also allows to estimate $\check{\mathbf{X}}$, i.e., the eigenvectors corresponding to the Q *smallest* eigenvalues of $\mathbf{R}_{yy}$. To achieve this, one should compute the columns of $\overline{\mathbf{X}}_q^{i+1}$ as the eigenvectors corresponding to the Q smallest eigenvalues of $\overline{\mathbf{R}}_q^i$.

Remark III) Reduced communication cost and computational complexity: Assuming $Q \ll M_k$, then the DACMEE algorithm has a reduced communication bandwidth at node k compared to the case where each node would transmit its raw sensor signal observations to an FC. Furthermore, the DACMEE algorithm often has a reduced computational complexity compared to the centralized EVD. This is due to the significant dimensionality reduction of the local EVD problems that are solved in each node. Indeed, the centralized EVD has a complexity of $O(M^3)$ per update, whereas the DACMEE algorithm has a complexity of $O\left((M_q + (K-1)Q)^3\right)$ at the updating node q. For example, if $K = 10$, $Q = 1$, and $M_k = 20$, $\forall\, k \in \mathcal{K}$, then the centralized EVD requires $\sim 64 \times 10^6$ flops to (re-)estimate $\mathbf{X}^i$, whereas the DACMEE algorithm requires only $\sim 24 \times 10^3$ flops per update. However, these numbers must be put into perspective, depending on the actual context in which these algorithms are applied, i.e., a static or an adaptive context. In an adaptive (tracking) context, where $\mathbf{X}^i$ is updated for each new block of N samples, we indeed find that the DACMEE algorithm is computationally cheaper than its centralized counterpart, but with the drawback of having a slower tracking performance due to its iterative nature.

In a static scenario, where only a single estimate of $\hat{\mathbf{X}}$ is computed, the computational complexity of the DACMEE algorithm should be multiplied with the number of iterations that are required to obtain a good estimate of $\hat{\mathbf{X}}$. It is noted that in the above example, the DACMEE algorithm can perform more than 1000 iterations before it reaches a similar computational complexity to its centralized counterpart, whereas it typically obtains a sufficiently accurate estimate of $\hat{\mathbf{X}}$ in much less iterations (see Section V).

Remark IV) Additional reduction of computational complexity: To reduce the per-node computational complexity even more, a node can reduce the dimensions of $\overline{\mathbf{R}}_q^i$ by using the sum of the broadcast signals, rather than incorporating each $\mathbf{z}_k^i$ separately in $\overline{\mathbf{R}}_q^i$. In this case, (24) is redefined as

$$\widetilde{\mathbf{y}}_k^i = \begin{bmatrix} \mathbf{y}_k \\ \sum_{q\in\mathcal{K}\setminus\{k\}} \mathbf{z}_q^i \end{bmatrix} \tag{39}$$

and several other variables ($\mathbf{\Lambda}_q^i$, $\mathbf{V}_q^i$, etc.) have to be redefined accordingly (details are omitted). Although this reduces the computational complexity at each node, it also reduces the degrees of freedom per updating step, yielding a slower convergence and hence slower adaptation in time-varying scenarios. It can be easily shown that all the convergence results in the sequel also hold for this simplification, based on similar convergence proofs. It is noted that the DACMEE algorithm in tree topology networks (see Section IV) will exploit a similar principle, i.e., the availability of summed $\mathbf{z}_k^i$-signal observations at each node.

Remark V) Implicit assumptions: It is noted that, for the time being, we have made 2 implicit assumptions (which are usually satisfied in practice) to guarantee that the DACMEE algorithm is well-defined:

1) The matrix $\mathbf{\Lambda}_q^i$ has full rank, i.e., $\mathbf{V}_q^i = \left(\mathbf{\Lambda}_q^i\right)^{-1}$ exists, $\forall\, i \in \mathbb{N}$, with q being the updating node in iteration i.
2) The $Q+1$ largest eigenvalues of $\overline{\mathbf{R}}_q^i$ are well-separated, i.e.,
$$\exists\, \epsilon > 0, \forall\, i \in \mathbb{N}, \forall\, n \in \{1,\ldots,Q\} : \; |\overline{\lambda}_n^i - \overline{\lambda}_{n+1}^i| > \epsilon \tag{40}$$
where $\overline{\lambda}_n^i$ is the n-th eigenvalue of $\overline{\mathbf{R}}_q^i$ with q being the updating node in iteration i.

This guarantees that $\mathbf{X}^{i+1}$ and $\mathbf{G}_{-q}$ are well-defined, i.e., there exists a unique $\overline{\mathbf{X}}_q^{i+1}$ in each iteration (up to a scaling ambiguity). It is noted that these assumptions are merely made for the sake of an easy exposition, and although they are mostly satisfied in practice, we briefly describe in Appendix VII-D how the algorithm can be modified in the rare cases where these assumptions are violated.

B. Convergence analysis

In this subsection, we prove a set of theorems that together imply convergence and optimality of the DACMEE algorithm. Before we proceed, it is noted that eigenvector estimation inherently has a scaling ambiguity with a certain scaling factor w where $|w| = 1$ (after normalization). To not overcomplicate the convergence statements and proofs, we will pragmatically ignore these ambiguities in the sequel. For example, with a slight abuse of notation, we will write $\lim_{i\to\infty} \mathbf{X}^i = \mathbf{X}^*$ actually meaning

$$\lim_{i\to\infty} \left(\min_{|w_q|=1,\, q=1,\ldots,Q} \|\mathbf{X}^i \mathrm{Diag}\left(w_1,\ldots,w_Q\right) - \mathbf{X}^*\|_F \right) = 0 \tag{41}$$

where $\|\cdots\|_F$ denotes the Frobenius norm. It is noted that this ambiguity may indeed hamper convergence (e.g., due to different scalings in different iterations). However, these can be easily eliminated by always selecting the proper unity-modulus scaling factors such that $\|\mathbf{X}^{i+1} - \mathbf{X}^i\|_F$ is minimized.

Besides the stationarity and ergodicity of the sensor signals (see Secion II), we make a couple of additional assumptions to make the theoretical analysis tractable:

- We assume that $\mathbf{R}_{yy}$ does not change over time.
- We will not incorporate the effect of estimation errors in $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i$, i.e., we implicitly assume that $N \to \infty$ in (25). The theoretical analysis is therefore only an approximation of the true behavior of the DACMEE algorithm with finite N.
- We assume ideal communication links (possibly relying on retransmission protocols in the case of link failures).
- We assume that the assumptions in Remark V hold. Although these are not crucial (see also Appendix VII-D), they substantially simplify the algorithm description/analysis.

In the theoretical analysis in the sequel, we implicitly assume that the above assumptions are satisfied, without repeating them in each of the listed theorems.

First, it is noted that each iteration of the original AO procedure, as described in the beginning of Subsection III-A, results in a monotonic increase of $J(\mathbf{X})$ under the constraint $\mathbf{X}^H\mathbf{X} = \mathbf{I}$. By construction, the DACMEE algorithm is equivalent to this AO procedure, and therefore we can state the following theorem:

Theorem III.1. *The objective function $J\left(\mathbf{X}\right)$ increases monotonically in each iteration of the DACMEE algorithm, i.e., $J\left(\mathbf{X}^{i+1}\right) \geq J\left(\mathbf{X}^i\right)$, $\forall\, i \in \mathbb{N}_0$. Furthermore, all points $\mathbf{X}^i$, $\forall\, i \in \mathbb{N}_0$, satisfy $\mathbf{X}^{i\,H}\mathbf{X}^i = \mathbf{I}_Q$.*

It is noted that $\mathbf{X}^0$ can be chosen randomly, and its columns are therefore not necessarily normalized. Therefore, it is possible that $J\left(\mathbf{X}^1\right) < J\left(\mathbf{X}^0\right)$. However, since the DACMEE algorithm performs an implicit normalization in each iteration, the monotonic increase holds for every $i > 1$.

Although Theorem III.1 guarantees the monotonic increase of $J\left(\mathbf{X}^i\right)$, this is no guarantee that $J\left(\mathbf{X}^i\right) \to J\left(\hat{\mathbf{X}}\right)$, let alone, that $\mathbf{X}^i \to \hat{\mathbf{X}}$. As a first step towards proving convergence of the DACMEE algorithm, we have the following theorem.

Theorem III.2. *The updates of the DACMEE algorithm satisfy*

$$\lim_{i\to\infty} \|\mathbf{X}^{i+1} - \mathbf{X}^i\|_F = 0\,. \tag{42}$$

Proof: See Appendix VII-A. ■

It is noted that (42) is a necessary condition for convergence of the DACMEE algorithm, but it is not a sufficient condition.

An equilibrium point of the DACMEE algorithm is defined as a point $\mathbf{X}^*$ such that, if $\mathbf{X}^i = \mathbf{X}^*$ at iteration $i = i^*$, then $\mathbf{X}^{j+1} = \mathbf{X}^j$, $\forall\, j \geq i^*$. An equilibrium is assumed to be unstable under the DACMEE algorithm update rules if a small perturbation on the equilibrium may cause the DACMEE algorithm to diverge away from the equilibrium. The following theorem addresses the equilibrium point(s) of the DACMEE algorithm and their stability.

Theorem III.3. *Let $\mathcal{X}^*$ denote the set of all equilibrium points of the DACMEE algorithm. Every $\mathbf{X}^* \in \mathcal{X}^*$ can only have eigenvectors of $\mathbf{R}_{yy}$ in its columns. Furthermore, $\mathcal{X}^*$ always contains $\hat{\mathbf{X}}$, as defined in (4), which is the only stable equilibrium point under the DACMEE update rules.*

Proof: See Appendix VII-B. ∎

It is noted that Theorem III.3 does not answer the question whether $\mathcal{X}^*$ is a singleton, i.e., whether $\hat{\mathbf{X}}$ is the only equilibrium point. The good news is that only eigenvectors of $\mathbf{R}_{yy}$ can form potential equilibria, and therefore it is highly unlikely that additional equilibria $\mathbf{X}^* \neq \hat{\mathbf{X}}$ exist, since this would imply that there exists an eigenvector that maximizes the objective function over K different constraint sets defined by (12)-(13), $\forall q \in \mathcal{K}$. Moreover, even when such a suboptimal equilibrium point would exists, it would be unstable.

Finally, we use the above theorems to show that the DACMEE algorithm indeed converges to an equilibrium point in $\mathcal{X}^*$.

Theorem III.4. *The limit $\lim_{i\to\infty} \mathbf{X}^i$ exists, i.e., the DACMEE algorithm converges.*

Proof: See Appendix VII-C. ∎

Due to inevitable estimation errors and numerical noise, we can safely assume that -in practice- the DACMEE algorithm will always diverge away from an unstable equilibrium point. Since $\hat{\mathbf{X}}$ is the only stable equilibrium point (see Theorem III.3), we finally conclude from Theorem III.4 that the DACMEE algorithm converges to the Q principal eigenvectors of $\mathbf{R}_{yy}$, i.e., $\lim_{i\to\infty} \mathbf{X}^i = \hat{\mathbf{X}}$.

IV. The DACMEE algorithm in networks with a tree topology

In this section, we explain how the DACMEE algorithm can be modified to operate in partially-connected networks, which are then assumed to be pruned to a tree topology (this will be motivated in Section IV-B). Before describing the algorithm, we first explain how the data flow is organized in such networks. For the sake of an easy exposition, we first focus on the data flow in star topology networks. This will provide us with the necessary tools to describe the DACMEE algorithm in tree topology networks.

In the sequel, we define $\mathcal{N}_k$ as the set of neighbors of node k, i.e., nodes that are connected to node k (node k itself excluded). The nodes with a single neighbor are referred to as leaf nodes.

A. Data flow in star topology networks

We first assume that the network has a star topology, where all the nodes are leaf nodes, except for a central node c for which $\mathcal{N}_c = \mathcal{K}\backslash\{c\}$. We assume that the central node c transmits (broadcasts) the same data to all the leaf nodes. We redefine the signal of which observations are broadcast by the central node c to all the leaf nodes as

$$\mathbf{z}_c^i = \mathbf{X}_c^{i\,H}\mathbf{y}_c + \sum_{l\in\mathcal{N}_c} \mathbf{z}_{lc}^i \tag{43}$$

where $\mathbf{z}_{lc}^i = \mathbf{X}_l^{i\,H}\mathbf{y}_l$ denotes the signal of which observations are transmitted from a leaf node l to the central node c. This definition also implies a causal relationship in the data exchange between the nodes, i.e., the leaf nodes first transmit observations of their respective $\mathbf{z}_{lc}^i$'s ($l \neq c$) to node c, after which node c computes the corresponding observations of $\mathbf{z}_c^i$ and broadcasts these to the leaf nodes.

It is important to note that the observations of $\mathbf{z}_c^i$ that are received at a leaf node l also contain a contribution from node l's own observations of $\mathbf{z}_{lc}^i$ (See (43)). This introduces a feedback path which affects the algorithm dynamics, as well as the equilibrium set (see also [11]). This feedback phenomenon in fact eliminates the monotonic increase of $J\left(\mathbf{X}^i\right)$.

Two different solutions have been described in [11] to tackle a similar feedback problem. The first solution is referred to as transmitter feedback cancellation (TFC) and matches well with point-to-point communication protocols, in which each connected node pair has a reserved communication link. In this case, the center node c can send different data to each of the leaf nodes. Let $\mathbf{z}_{ck}^i$ denote the signal of which node c sends observations to node k, then the feedback path is removed by setting

$$\mathbf{z}_{ck}^i = \mathbf{X}_c^{i\,H}\mathbf{y}_c + \sum_{l\in\mathcal{N}_c\backslash\{k\}} \mathbf{z}_{lc}^i\,. \tag{44}$$

The second solution is referred to as receiver feedback cancellation (RFC), where the central node still broadcasts the same data to all the leaf nodes [11]. In this case, the leaf nodes remove their own contribution from the observations of $\mathbf{z}_c^i$, i.e., node k effectively uses the feedback-corrected input signal

$$\mathbf{z}_{c,-k}^i = \mathbf{z}_c^i - \mathbf{z}_{kc}^i\,. \tag{45}$$

RFC is clearly much more efficient in terms of communication bandwidth.

It is noted that $\mathbf{z}_{c,-k}^i = \mathbf{z}_{ck}^i$, i.e., the DACMEE algorithm behaves exactly the same, whether we apply TFC or RFC. Therefore, we do not have to distinguish between both cases in the sequel, i.e., we will only focus on TFC for the sake of an easier exposition.

B. Data flow in tree topology networks

Let $\mathbf{z}_{kq}^i$ denote the signal of which observations are transmitted from node k to node q (we do not distinguish between leaf nodes and a central node). The definition of the $\mathbf{z}_{kq}^i$ signals within a tree topology, as well as the corresponding data flow,

are similarly defined as in (44), here repeated for convenience:

$$\mathbf{z}_{kq}^i = \mathbf{X}_k^{i\,H}\mathbf{y}_k + \sum_{l\in\mathcal{N}_k\backslash\{q\}} \mathbf{z}_{lk}^i \ . \tag{46}$$

The motivation for pruning the network to a tree topology, is based on the observations in Section IV-A, and can be summarized as follows:

1) Tree topologies have no cycles, and therefore all possible feedback paths are eliminated when using RFC or TFC. As explained earlier, feedback paths severely affect the DACMEE algorithm dynamics, as well as the equilibrium set.
2) In a tree topology, there is a natural order to compute the observations of the different $\mathbf{z}_{kq}^i$'s as defined in (46). This is not the case in ad hoc networks, where deadlock situations may arise.

In a tree topology, the computation of the $\mathbf{z}_{kq}^i$'s and the corresponding data flow can be completely data-driven without any central coordination: a node k computes and transmits N observations of $\mathbf{z}_{kq}^i$ as soon as it has received N observations of $\mathbf{z}_{lk}^i$ from its neighbors $l \in \mathcal{N}_k\backslash\{q\}$. Since any leaf node k only has one neighbor (say, q), $\mathbf{z}_{kq}^i = \mathbf{X}_k^{i\,H}\mathbf{y}_k$, which does not rely on observations from any other node, and therefore the leaf nodes will initiate the computation of (46). This data-driven approach will naturally result in a so-called 'fusion flow' from the leaf nodes to the root node(s), followed by a so-called 'diffusion flow' from the root node(s) to the leaf nodes (see [11]).

C. The DACMEE algorithm in tree topology networks

The vector $\widetilde{\mathbf{y}}_k$, which contains all the signals of which observations are available to node k, is redefined as (compare with (24))

$$\widetilde{\mathbf{y}}_k^i = \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{\rightarrow k}^i \end{bmatrix} \tag{50}$$

where $\mathbf{z}_{\rightarrow k}^i$ is the stacked version of all the signals $\mathbf{z}_{qk}^i$ for $q \in \mathcal{N}_k$, where the $\mathbf{z}_{qk}^i$'s are ordered such that $\mathbf{z}_{mk}$ is above $\mathbf{z}_{lk}$ if $m < l$. In the sequel, we will always use the same order whenever we stack variables that relate to the different neighbors of node k. Similar to (46), we also define

$$\mathbf{D}_{kq}^i = \mathbf{X}_k^{i\,H}\mathbf{X}_k^i + \sum_{l\in\mathcal{N}_k\backslash\{q\}} \mathbf{D}_{lk}^i \tag{51}$$

and its square root

$$\mathbf{L}_{kq}^i = \left(\mathbf{D}_{kq}^i\right)^{\frac{1}{2}} \ . \tag{52}$$

With this, we define

$$\mathbf{\Lambda}_{\rightarrow k}^i = \text{Blkdiag}\left(\mathbf{I}_{M_k}, \mathbf{L}_{l_1 k}^i, \ldots, \mathbf{L}_{l_{n_k} k}^i\right) \tag{53}$$

with $n_k = |\mathcal{N}_k|$ and $\{l_1, \ldots, l_{n_k}\} = \mathcal{N}_k$, and where we use the same order as the $\mathbf{z}_{qk}^i$'s in $\mathbf{z}_{\rightarrow k}^i$, i.e., $l_j > l_{j-1}$. Finally, we define $\mathcal{V}_{kq}$ as the set of nodes in the tree branch that would be disconnected from the rest of the tree if the link between nodes k and q is cut, and where $k \in \mathcal{V}_{kq}$ and $q \notin \mathcal{V}_{kq}$. Notice that

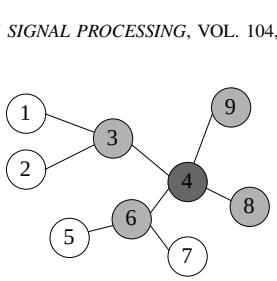

Fig. 2. Example of a network graph with tree topology with 9 sensor nodes.

$$\mathbf{z}_{kq}^i = \sum_{l\in\mathcal{V}_{kq}} \mathbf{X}_l^{i\,H}\mathbf{y}_l^i \tag{54}$$

and

$$\mathbf{D}_{kq}^i = \sum_{l\in\mathcal{V}_{kq}} \mathbf{X}_l^{i\,H}\mathbf{X}_l^i \ . \tag{55}$$

Example: Consider the network graph depicted in Fig. 2, and consider nodes 3 and 4 in particular. It holds that $\mathcal{V}_{34} = \{1, 2, 3\}$ and $\mathcal{V}_{43} = \{4, 5, 6, 7, 8, 9\}$. The observations that node 3 transmits to node 4 correspond to (see (46))

$$\mathbf{z}_{34}^i = \mathbf{X}_3^{i\,H}\mathbf{y}_3 + \mathbf{z}_{13}^i + \mathbf{z}_{23}^i$$

and

$$\mathbf{z}_{\rightarrow 4}^i = \begin{bmatrix} \mathbf{z}_{34}^i \\ \mathbf{z}_{64}^i \\ \mathbf{z}_{84}^i \\ \mathbf{z}_{94}^i \end{bmatrix} \ .$$

Using (51) or (55), we find that

$$\begin{aligned} \mathbf{D}_{34}^i &= \mathbf{X}_3^{i\,H}\mathbf{X}_3^i + \mathbf{D}_{13}^i + \mathbf{D}_{23}^i \\ &= \mathbf{X}_1^{i\,H}\mathbf{X}_1^i + \mathbf{X}_2^{i\,H}\mathbf{X}_2^i + \mathbf{X}_3^{i\,H}\mathbf{X}_3^i \end{aligned}$$

$$\begin{aligned} \mathbf{D}_{43}^i &= \mathbf{X}_4^{i\,H}\mathbf{X}_4^i + \mathbf{D}_{64}^i + \mathbf{D}_{84}^i + \mathbf{D}_{94}^i \\ &= \sum_{j=4}^{9} \mathbf{X}_j^{i\,H}\mathbf{X}_j^i \end{aligned}$$

(note that a similar relationship holds for $\mathbf{z}_{34}^i$ and $\mathbf{z}_{43}^i$ based on (46) and (54)). We also observe that

$$\mathbf{\Lambda}_{\rightarrow 4}^i = \text{Blkdiag}\left(\mathbf{I}_{M_4}, \mathbf{L}_{34}^i, \mathbf{L}_{64}^i, \mathbf{L}_{84}^i, \mathbf{L}_{94}^i\right) \ .$$

The description of the DACMEE algorithm for tree topology networks is given in Table II. Similar to the DACMEE algorithm in fully-connected networks, the transmission of the $\mathbf{G}$ and $\mathbf{D}$ parameters is assumed to be negligible compared to the transmission of the signal observations between nodes, assuming $N \gg Q^2$ (see also Remark IV). We also reiterate that the communication bandwidth can be significantly reduced by using local broadcasts and RFC signals instead of the TFC signals defined in (46).

The convergence analysis of the DACMEE algorithm for fully-connected networks can be easily generalized to the case of tree topology networks, i.e., the theorems in Section III-B also hold for tree-topology networks (details omitted).

Remark VI: It is noted that each node can compute observations of the PCA- or KLT-based compressed signal $\hat{\mathbf{z}} = \hat{\mathbf{X}}^H\mathbf{y}$

TABLE II
DESCRIPTION OF THE DACMEE ALGORITHM IN A NETWORK WITH A TREE TOPOLOGY.

DACMEE algorithm in a tree topology network

1) Set $i \leftarrow 0$, $q \leftarrow 1$, and initialize all $\mathbf{X}_k^0$, $\forall\, k \in \mathcal{K}$, with random entries.
2) Each node $k \in \mathcal{K}$ transmits $\mathbf{D}_{kl}^i$ to node l, $\forall\, l \in \mathcal{N}_k$, based on (51). The order in which these are computed and forwarded is dictated by (51).
3) Each node $k \in \mathcal{K}$ transmits N observations of $\mathbf{z}_{kl}^i$ to node l, $\forall\, l \in \mathcal{N}_k$, based on (46). The order in which these are computed and forwarded by the different nodes is dictated by (46).
4) At node q:
   - Compute $\mathbf{R}_{\tilde{y}_q \tilde{y}_q}^i$ based on (25), using the definition of $\widetilde{\mathbf{y}}_k^i$ given by (50).
   - Compute the square root factorization $\mathbf{L}_{lq}^{i+1} = \left(\mathbf{D}_{lq}^{i+1}\right)^{\frac{1}{2}}$, $\forall\, l \in \mathcal{N}_q$.
   - Construct the block-diagonal matrix $\mathbf{\Lambda}_{\rightarrow q}^i$ as defined in (53) and compute the inverse of each diagonal block to construct the block-diagonal matrix $\mathbf{V}_{\rightarrow q}^i = \left(\mathbf{\Lambda}_{\rightarrow q}^i\right)^{-1}$.
   - Compute the Q principal eigenvectors of the matrix
   $$\overline{\mathbf{R}}_q^i = \mathbf{V}_{\rightarrow q}^{i\,H} \mathbf{R}_{\tilde{y}_q \tilde{y}_q}^i \mathbf{V}_{\rightarrow q}^i \tag{47}$$
   and construct $\overline{\mathbf{X}}_q^{i+1}$ with the eigenvectors in the columns, sorted in decreasing order of eigenvalue magnitudes.
   - Set
   $$\begin{bmatrix} \mathbf{X}_q^{i+1} \\ \mathbf{G}_{\rightarrow q} \end{bmatrix} = \mathbf{V}_{\rightarrow q}^i \overline{\mathbf{X}}_q^{i+1} \tag{48}$$
5) Define the partitioning $\mathbf{G}_{\rightarrow q} = \begin{bmatrix} \mathbf{G}_{l_1}^T \ldots \mathbf{G}_{l_{n_q}}^T \end{bmatrix}^T$ where each $\mathbf{G}_{l_k}$ is a $Q \times Q$ matrix and where $\{l_1, \ldots, l_{n_q}\} = \mathcal{N}_q$. Disseminate $\mathbf{G}_l$ over the tree branch $\mathcal{V}_{lq}$, $\forall\, l \in \mathcal{N}_q$. Each node $k \in \mathcal{V}_{lq}$ updates
   $$\mathbf{X}_k^{i+1} = \mathbf{X}_k^i \mathbf{G}_l\ . \tag{49}$$
6) $i \leftarrow i+1$ and $q \leftarrow (q \mod K) + 1$.
7) Return to step 2.

(see Section II-B). Indeed, for each node $k \in \mathcal{K}$, it holds that

$$\lim_{i \to \infty} \left(\mathbf{X}_k^{i\,H} \mathbf{y}_k + \sum_{q \in \mathcal{N}_k} \mathbf{z}_{qk}^i \right) = \hat{\mathbf{z}} \tag{56}$$

and therefore each node can reconstruct the observations of the network-wide vector $\mathbf{y}$ using the approximation $\mathbf{y} \approx \hat{\mathbf{X}}\hat{\mathbf{z}}$. This decompression requires some minor additional data exchange such that each node has access to the complete $\mathbf{X}^i$.

V. SIMULATIONS

A. Generation of sensor signal observations

In this section, we provide Monte-Carlo (MC) simulations of the DACMEE algorithm[7]. In each MC run, a new scenario is created with K nodes (for different values of K), each collecting observations of a different 15-dimensional stochastic vector $\mathbf{y}_k$, $\forall\, k \in \mathcal{K}$, where the observations of the stacked vector $\mathbf{y}$ are generated as

$$\mathbf{y}[t] = \mathbf{A}\mathbf{d}[t] + \mathbf{n}[t] \tag{57}$$

where $\mathbf{A}$ is a fixed (independent from t) $15K \times 10$ matrix of which the entries are randomly drawn from a uniform distribution over the interval $[-0.5; 0.5]$, $\mathbf{d}[t]$ is an observation of a 10-dimensional stochastic vector of which the entries are independent and uniformly distributed over the interval $[-0.5; 0.5]$ and $\mathbf{n}[t]$ is an observation of a $15K$-dimensional stochastic vector of which the entries are independent and uniformly distributed over the interval $[-\sqrt{0.1}/2; \sqrt{0.1}/2]$ (modelling spatially uncorrelated sensor noise). The number of observations of $\mathbf{y}[t]$ is set to $N = 10000$ in each MC run.

B. Simulation results

1) Fully-connected networks: The upper plot in Fig. 3 demonstrates the monotonic increase of $J(\mathbf{X}^i)$ (averaged over 200 MC runs) over the iterations of the DACMEE algorithm in fully-connected WSNs for $K = 10$ nodes and for different values of Q. The objective function converges to the optimal value $J(\hat{\mathbf{X}})$ (denoted by the dashed lines). The lower plot shows the mean squared error (MSE) of the entries of $\mathbf{X}^i$ compared to $\hat{\mathbf{X}}$, i.e.,

$$\text{MSE}^i = \frac{1}{MQ} Tr\left\{ \left(\mathbf{X}^i - \hat{\mathbf{X}}\right)^H \left(\mathbf{X}^i - \hat{\mathbf{X}}\right) \right\}\ . \tag{58}$$

[7]Matlab code to reproduce the simulation results in this section can be found at *http://homes.esat.kuleuven.be/~abertran/software.html*

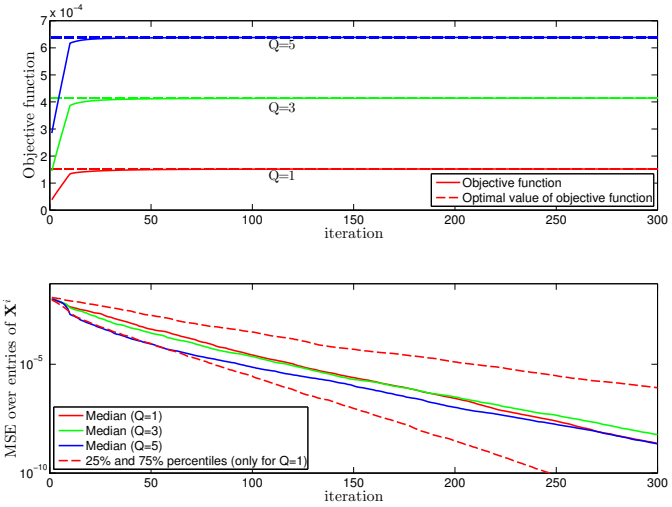

Fig. 3. Convergence properties of the DACMEE algorithm in a fully-connected WSN with $K = 10$ nodes and for different values of Q.

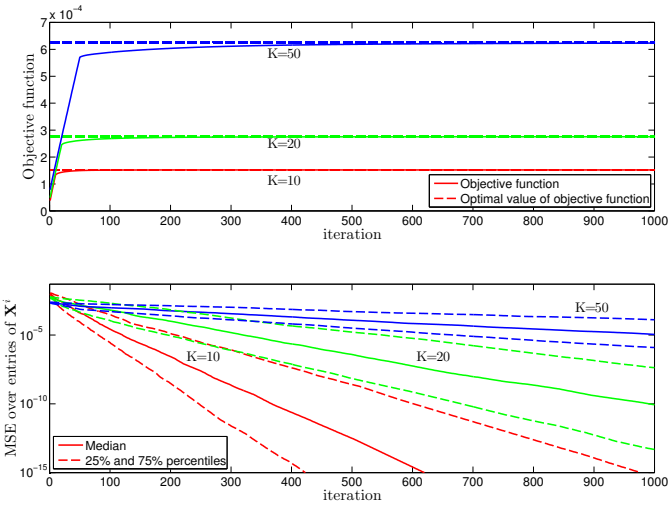

Fig. 4. Convergence properties of the DACMEE algorithm in fully-connected WSNs for different network sizes K and for $Q = 1$.

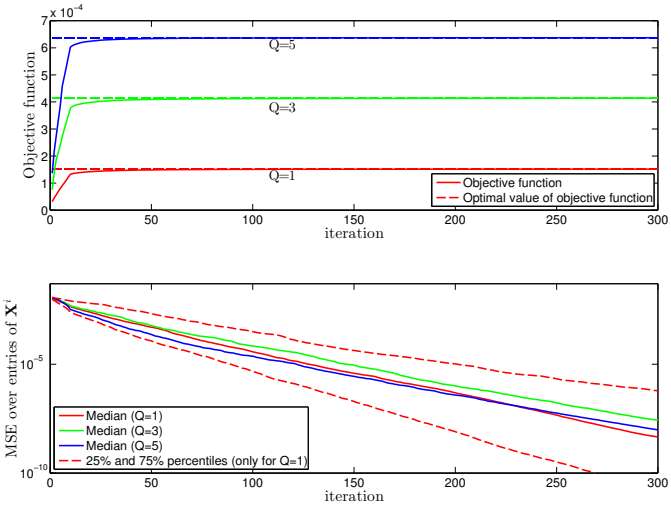

Fig. 5. Convergence properties of the DACMEE algorithm in WSNs with a tree topology with $K = 10$ nodes and for different values of Q.

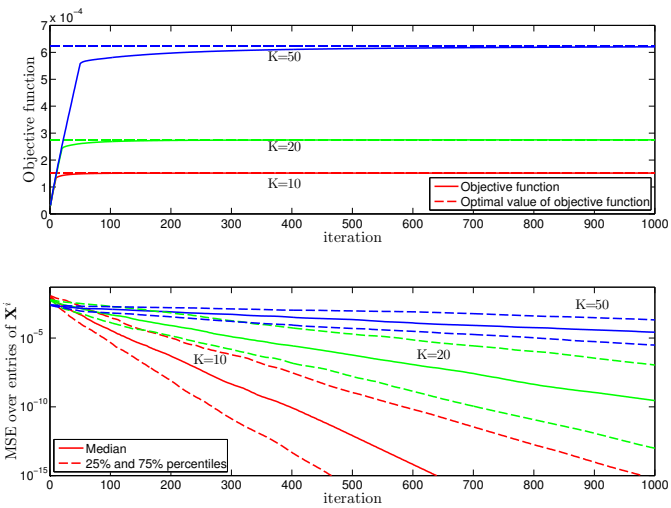

Fig. 6. Convergence properties of the DACMEE algorithm in WSNs with a tree topology for different network sizes K and for $Q = 1$.

The figure shows the median (full line), as well as the 25% and 75% percentiles (dashed lines) over the 200 MC runs. These percentiles are only shown for $Q = 1$ for the sake of intelligibility, since the percentile boundaries of $Q = 3$ and $Q = 5$ are not very different from those of $Q = 1$. The main conclusion that can be drawn from Fig. 3 is that the convergence speed does not significantly depend on Q.

Fig. 4 shows the same metrics, this times for different network sizes, i.e., $K = 10$, $K = 20$ and $K = 50$, and where Q is fixed to $Q = 1$. Since the the nodes are updated in a sequential round-robin fashion, it is not surprising that the number of nodes K has a direct influence on the convergence speed.

2) Networks with a tree topology: Fig. 5 and Fig. 6 show the same metrics for the case where the DACMEE algorithm is operated in WSNs with a tree topology, where a different random tree is constructed in each individual MC run. Despite the fact that the updating node in a tree topology has much less degrees of freedom compared to the fully-connected case (compare the dimensions of (35) and (48)), the convergence is not much slower in a tree topology. However, the convergence speed may depend on the scenario and the signal content, and therefore one should be careful to generalize this conclusion.

VI. Conclusions

We have described a distributed adaptive algorithm, referred to as the DACMEE algorithm, to estimate the eigenvectors corresponding to the Q largest or smallest eigenvalues of the network-wide sensor signal covariance matrix in a WSN. It has been demonstrated that the eigenvectors can be estimated without the need to collect all the sensor signal observations in an FC to explicitly construct the network-wide covariance matrix. We have first described the DACMEE algorithm for fully-connected networks, and we have then extended it for application in networks with a tree topology. We have provided a theoretical convergence analysis, which has been validated

by means of numerical simulations. We have also briefly illustrated the use of the DACMEE algorithm in a context of distributed PCA.

VII. Appendix

A. Proof of Theorem III.2

Proof outline: We will first prove that the monotonic increase of $J(\mathbf{X})$ as addressed in Theorem III.1 also holds for the first diagonal element of $\mathbf{X}^H\mathbf{R}_{yy}\mathbf{X}$, i.e.,

$$f(\mathbf{x}_1^{i+1}) \geq f(\mathbf{x}_1^i) \tag{59}$$

where $f(\mathbf{x}) = \mathbf{x}^H\mathbf{R}_{yy}\mathbf{x}$ and where $\mathbf{x}_n^i$ denotes the n-th column of $\mathbf{X}^i$. Using this fact, we will show that the theorem holds for the first column of $\mathbf{X}^i$, i.e., $\lim_{i\to\infty}\|\mathbf{x}_1^{i+1}-\mathbf{x}_1^i\| = 0$. We will then explain how the latter also implies that the same holds for the second column, and by an induction argument for all Q columns of $\mathbf{X}^i$.

Proof: Assume that node q is the updating node at iteration i, and let $\overline{\mathbf{x}}_n^{i+1}$ denote the n-th column of $\overline{\mathbf{X}}_q^{i+1}$ as defined in Table I. Since $\overline{\mathbf{X}}_q^{i+1}$ contains the Q principal eigenvectors of the matrix $\overline{\mathbf{R}}_q^i$ defined in (34), we have that

$$\overline{\mathbf{x}}_1^{i+1} = \arg\max_{\|\overline{\mathbf{x}}\|=1} \overline{\mathbf{x}}^H\overline{\mathbf{R}}_q^i\overline{\mathbf{x}}\,. \tag{60}$$

From (60), and based on a similar reasoning as in Section III-A, i.e., the equivalence between the optimization problems (11)-(13) and (31)-(32) with objective function J replaced by $f(\mathbf{X}\mathbf{e}_1)$, where $\mathbf{e}_1 = [1\ 0\ \ldots\ 0]^T$, we find that the first column of $\mathbf{X}^{i+1}$, as computed by the DACMEE algorithm also maximizes f under the constraints (12)-(13). Therefore, and since $\mathbf{X}^i$ from the previous iteration also satisfies the constraints (12)-(13), we have that

$$f(\mathbf{x}_1^{i+1}) \geq f(\mathbf{x}_1^i),\ \forall\, i \in \mathbb{N}\,. \tag{61}$$

Since the sequence $\left\{f(\mathbf{x}_1^i)\right\}_{i\in\mathbb{N}}$ is monotonically increasing and since it has an upper bound,

$$\lim_{i\to\infty}\left(f(\mathbf{x}_1^{i+1}) - f(\mathbf{x}_1^i)\right) = 0\,. \tag{62}$$

Let us now consider the EVD of $\overline{\mathbf{R}}_q^i$ defined as

$$\overline{\mathbf{R}}_q^i = \overline{\mathbf{U}}_q^i\overline{\mathbf{\Sigma}}_q^i\overline{\mathbf{U}}_q^{i\,H}\,. \tag{63}$$

Then for any vector $\overline{\mathbf{x}}$,

$$\overline{\mathbf{x}}^H\overline{\mathbf{R}}_q^i\overline{\mathbf{x}} = \sum_{j=1}^{M_q+(K-1)Q}\left(\overline{\mathbf{x}}^H\overline{\mathbf{u}}_j\right)^2\overline{\lambda}_j \tag{64}$$

where $\overline{\lambda}_1 \geq \overline{\lambda}_2 \geq \ldots$ are the eigenvalues in the diagonal elements of $\overline{\mathbf{\Sigma}}_q^i$ and where the $\overline{\mathbf{u}}_j$'s are the corresponding eigenvectors in $\overline{\mathbf{U}}_q^i$. Since $\overline{\mathbf{U}}_q^i$ is a unitary matrix, we have that

$$\|\overline{\mathbf{x}}\| = 1 \Rightarrow \overline{\mathbf{x}}^H\overline{\mathbf{R}}_q^i\overline{\mathbf{x}} \leq \left|\overline{\mathbf{x}}^H\overline{\mathbf{u}}_1\right|^2\overline{\lambda}_1 + \left(1-\left|\overline{\mathbf{x}}^H\overline{\mathbf{u}}_1\right|^2\right)\overline{\lambda}_2\,. \tag{65}$$

Since $\overline{\mathbf{X}}_q^{i+1}$ contains the Q principal eigenvectors of $\overline{\mathbf{R}}_q^i$ in its columns (see Table I), we have that

$$\overline{\mathbf{x}}_n^{i+1} = \overline{\mathbf{u}}_n\,,\ n = 1\ldots Q \tag{66}$$

$$\overline{\mathbf{x}}_n^{i+1\,H}\overline{\mathbf{R}}_q^i\overline{\mathbf{x}}_n^{i+1} = \overline{\lambda}_n\,,\ n = 1\ldots Q\,. \tag{67}$$

Furthermore, we define $\mathbf{p}_n^i$ as the n-th column of the matrix $\mathbf{P}_q^i$, which is defined as

$$\mathbf{P}_q^i = \mathbf{\Lambda}_q^i\begin{bmatrix}\mathbf{X}_q^i\\ \mathbf{I}_Q\\ \vdots\\ \mathbf{I}_Q\end{bmatrix} \tag{68}$$

where $\mathbf{\Lambda}_q^i$ was defined in (28). From (26)-(28) and since $\mathbf{X}^{i\,H}\mathbf{X}^i = \mathbf{I}_Q$, $\forall\, i \in \mathbb{N}$, we have that

$$\mathbf{P}_q^{i\,H}\mathbf{P}_q^i = \mathbf{I}_Q,\ \forall\, i \in \mathbb{N}\,, \tag{69}$$

which shows that the $\mathbf{p}_n^i$'s have unity norm. Therefore, we can set $\overline{\mathbf{x}} = \mathbf{p}_1^i$ in (65) and use (66) to obtain

$$\mathbf{p}_1^{i\,H}\overline{\mathbf{R}}_q^i\mathbf{p}_1^i \leq \left|\mathbf{p}_1^{i\,H}\overline{\mathbf{x}}_1^{i+1}\right|^2\overline{\lambda}_1 + \left(1-\left|\mathbf{p}_1^{i\,H}\overline{\mathbf{x}}_1^{i+1}\right|^2\right)\overline{\lambda}_2\,. \tag{70}$$

Using this with (67) and (40), we find

$$\overline{\mathbf{x}}_1^{i+1\,H}\overline{\mathbf{R}}_q^i\overline{\mathbf{x}}_1^{i+1} - \mathbf{p}_1^{i\,H}\overline{\mathbf{R}}_q^i\mathbf{p}_1^i \geq \left(1-\left|\mathbf{p}_1^{i\,H}\overline{\mathbf{x}}_1^{i+1}\right|^2\right)\left(\overline{\lambda}_1-\overline{\lambda}_2\right) \tag{71}$$

$$> \left(1-\left|\mathbf{p}_1^{i\,H}\overline{\mathbf{x}}_1^{i+1}\right|^2\right)\epsilon\,. \tag{72}$$

From (34), (68), and the fact that $\mathbf{V}_q^i = \left(\mathbf{\Lambda}_q^i\right)^{-1}$ (see (29)), we find that

$$\mathbf{P}^{i\,H}\overline{\mathbf{R}}_q^i\mathbf{P}^i = \left[\mathbf{X}_q^{i\,H}\,\mathbf{I}_Q\ \ldots\ \mathbf{I}_Q\right]\mathbf{R}_{\tilde{y}_q\tilde{y}_q}^i\begin{bmatrix}\mathbf{X}_q^i\\ \mathbf{I}_Q\\ \vdots\\ \mathbf{I}_Q\end{bmatrix}\,. \tag{73}$$

From the definition of $\mathbf{C}_k^i$ in (14)-(16), we find that $\mathbf{C}_k^i\left[\mathbf{X}_q^{i\,T}\,\mathbf{I}_Q\ \ldots\ \mathbf{I}_Q\right]^T = \mathbf{X}^i$. Using this with (20) in (73) therefore yields

$$\mathbf{P}^{i\,H}\overline{\mathbf{R}}_q^i\mathbf{P}^i = \mathbf{X}^{i\,H}\mathbf{R}_{yy}\mathbf{X}^i\,. \tag{74}$$

or equivalently for column n

$$\mathbf{p}_n^{i\,H}\overline{\mathbf{R}}_q^i\mathbf{p}_n^i = \mathbf{x}_n^{i\,H}\mathbf{R}_{yy}\mathbf{x}_n^i = f(\mathbf{x}_n^i)\,. \tag{75}$$

By plugging this into (72), and using the fact that

$$\overline{\mathbf{x}}_1^{i+1\,H}\overline{\mathbf{R}}_q^i\overline{\mathbf{x}}_1^{i+1} = \mathbf{x}_1^{i+1\,H}\mathbf{R}_{yy}\mathbf{x}_1^{i+1} = f(\mathbf{x}_1^{i+1}) \tag{76}$$

we find that

$$f(\mathbf{x}_1^{i+1}) - f(\mathbf{x}_1^i) > \left(1-\left|\mathbf{p}_1^{i\,H}\overline{\mathbf{x}}_1^{i+1}\right|^2\right)\epsilon\,. \tag{77}$$

Note that this holds for any iteration i and for any arbitrary updating node q in the DACMEE algorithm. Therefore, and because of (62) and the fact that the righthand side of (77) is non-negative, it follows from the sandwich theorem that

$$\lim_{i\to\infty}\left|\mathbf{p}_1^{i\,H}\overline{\mathbf{x}}_1^{i+1}\right|^2 = 1 \tag{78}$$

or equivalently

$$\lim_{i\to\infty} \|\overline{\mathbf{x}}_1^{i+1} - \mathbf{p}_1^i\| = 0 \,. \tag{79}$$

Define $\mathbf{e}_n$ as a selection vector which has only zero-valued entries except for the n-th entry which is set to 1. With this, we can rewrite (79) as

$$\lim_{i\to\infty} \| \left(\overline{\mathbf{X}}_q^{i+1} - \mathbf{P}_q^i \right) \mathbf{e}_1 \| = 0 \,. \tag{80}$$

This means that we can replace the first column of $\overline{\mathbf{X}}_q^{i+1}$ by the first column of $\mathbf{P}_q^i$ in (35) when $i \to \infty$, i.e.,

$$\lim_{i\to\infty} \left\| \left(\left[\begin{array}{c} \mathbf{X}_q^{i+1} \\ \mathbf{G}_{-q} \end{array} \right] - \mathbf{V}_q^i \mathbf{P}_q^i \right) \mathbf{e}_1 \right\| = 0 \,. \tag{81}$$

When substituting (68), and using (29), we find that

$$\lim_{i\to\infty} \left\| \left(\left[\begin{array}{c} \mathbf{X}_q^{i+1} \\ \mathbf{G}_{-q} \end{array} \right] - \left[\begin{array}{c} \mathbf{X}_q^{i} \\ \mathbf{I}_Q \\ \vdots \\ \mathbf{I}_Q \end{array} \right] \right) \mathbf{e}_1 \right\| = 0 \,. \tag{82}$$

Using (38), we eventually find that

$$\lim_{i\to\infty} \left\| \left(\mathbf{X}^{i+1} - \mathbf{X}^i \right) \mathbf{e}_1 \right\| = \left\| \mathbf{x}_1^{i+1} - \mathbf{x}_1^i \right\| = 0 \tag{83}$$

which proves the theorem for the first column of $\mathbf{X}^i$. The proof for the other columns then relies on an induction argument. To illustrate this, we prove the theorem for the second column of $\mathbf{X}^i$. We first show that (62) also holds for $\mathbf{x}_2^i$. To this end, we define

$$g\left(\mathbf{x}_1, \mathbf{x}_2\right) = f(\mathbf{x}_1) + f(\mathbf{x}_2) \,. \tag{84}$$

Since $\overline{\mathbf{X}}_q^{i+1}$ contains the Q principal eigenvectors of the matrix $\overline{\mathbf{R}}_q^i$ defined in (34), we have that

$$\left[\overline{\mathbf{x}}_1^{i+1}\, \overline{\mathbf{x}}_2^{i+1}\right] = \underset{\|\overline{\mathbf{x}}_1\|=\|\overline{\mathbf{x}}_2\|=1,\, \overline{\mathbf{x}}_1^H \overline{\mathbf{x}}_2 = 0}{\arg\max} \operatorname{Tr}\left\{ \left[\overline{\mathbf{x}}_1\, \overline{\mathbf{x}}_2\right]^H \overline{\mathbf{R}}_q^i \left[\overline{\mathbf{x}}_1\, \overline{\mathbf{x}}_2\right] \right\} \,. \tag{85}$$

From (85), and based on a similar reasoning as in Section III-A, i.e., the equivalence between the optimization problems (11)-(13) and (31)-(32) with objective function J replaced by $g(\mathbf{X}\mathbf{e}_1, \mathbf{X}\mathbf{e}_2)$, we find that the DACMEE algorithm also maximizes g under the constraints (12)-(13). Since $\mathbf{X}^i$ also satisfies the constraints (12)-(13),

$$g(\mathbf{x}_1^{i+1}, \mathbf{x}_2^{i+1}) \geq g(\mathbf{x}_1^i, \mathbf{x}_2^i), \;\; \forall\, i \in \mathbb{N} \,. \tag{86}$$

Since the sequence $\left\{ g(\mathbf{x}_1^i, \mathbf{x}_2^i) \right\}_{i\in\mathbb{N}}$ is monotonically increasing and since it has an upper bound,

$$\lim_{i\to\infty} \left(g(\mathbf{x}_1^{i+1}, \mathbf{x}_2^{i+1}) - g(\mathbf{x}_1^i, \mathbf{x}_2^i) \right) = 0 \tag{87}$$

and with (84) and (62)

$$\lim_{i\to\infty} \left(f(\mathbf{x}_2^{i+1}) - f(\mathbf{x}_2^i) \right) = 0 \,. \tag{88}$$

Because of (78) and since (69) implies that ${\mathbf{p}_2^i}^H \mathbf{p}_1^i = 0$, $\forall\, i \in \mathbb{N}$,

$$\lim_{i\to\infty} {\mathbf{p}_2^i}^H \overline{\mathbf{x}}_1^{i+1} = 0 \,. \tag{89}$$

Furthermore, (65) can be extended to

$$\begin{aligned} \|\overline{\mathbf{x}}\| = 1 \Rightarrow \overline{\mathbf{x}}^H \overline{\mathbf{R}}_q^i \overline{\mathbf{x}} \leq \left|\overline{\mathbf{x}}^H \overline{\mathbf{u}}_1\right|^2 \overline{\lambda}_1 + \left|\overline{\mathbf{x}}^H \overline{\mathbf{u}}_2\right|^2 \overline{\lambda}_2 \\ + \left(1 - \left|\overline{\mathbf{x}}^H \overline{\mathbf{u}}_1\right|^2 - \left|\overline{\mathbf{x}}^H \overline{\mathbf{u}}_2\right|^2\right) \overline{\lambda}_3 \,. \end{aligned} \tag{90}$$

By setting $\overline{\mathbf{x}} = \mathbf{p}_2^i$ in (65) and using (66), we obtain

$$\begin{aligned} {\mathbf{p}_2^i}^H \overline{\mathbf{R}}_q^i \mathbf{p}_2^i \leq \left|{\mathbf{p}_2^i}^H \overline{\mathbf{x}}_1^{i+1}\right|^2 \overline{\lambda}_1 + \left|{\mathbf{p}_2^i}^H \overline{\mathbf{x}}_2^{i+1}\right|^2 \overline{\lambda}_2 \\ + \left(1 - \left|{\mathbf{p}_2^i}^H \overline{\mathbf{x}}_1^{i+1}\right|^2 - \left|{\mathbf{p}_2^i}^H \overline{\mathbf{x}}_2^{i+1}\right|^2\right) \overline{\lambda}_3 \,. \end{aligned} \tag{91}$$

From (89), we know that

$$i \to \infty : {\mathbf{p}_2^i}^H \overline{\mathbf{R}}_q^i \mathbf{p}_2^i \leq \left|{\mathbf{p}_2^i}^H \overline{\mathbf{x}}_2^{i+1}\right|^2 \overline{\lambda}_2 + \left(1 - \left|{\mathbf{p}_2^i}^H \overline{\mathbf{x}}_2^{i+1}\right|^2\right) \overline{\lambda}_3 \tag{92}$$

which has a similar form to (70). Therefore, we can use a similar reasoning as before to show that $\lim_{i\to\infty} \|\mathbf{x}_2^{i+1} - \mathbf{x}_2^i\| = 0$, and with a simple induction argument, we eventually find that

$$\lim_{i\to\infty} \|\mathbf{x}_n^{i+1} - \mathbf{x}_n^i\| = 0 \,, \;\; n = 1 \ldots Q \tag{93}$$

which proves the theorem.

B. Proof of Theorem III.3

Proof outline: The proof contains three parts. In the first part, we will show that any equilibrium point of the DACMEE algorithm should only contain eigenvectors of $\mathbf{R}_{yy}$, based on the following steps:

- We will first derive a specific set of equations (SOE) that satisfy the update rules of the DACMEE algorithm for an updating node $q \in \mathcal{K}$. This SOE then contains variables that depend on $\mathbf{X}^{i+1}$ (after the update), and $\mathbf{X}^i$ (before the update).
- We will then assume that $\mathbf{X}^i$ is an equilibrium point, such that we can set $\mathbf{X}^{i+1} = \mathbf{X}^i = \mathbf{X}^*$, in this system of equations.
- By setting up similar SOEs for the updates at all the other nodes (each time assuming an equilibrium point), we obtain a new combined SOE that can be shown to be independent of the updating nodes.
- Since $\mathbf{X}^i$ is assumed to be an equilibrium point, this final SOE will provide a necessary condition for an equilibrium point $\mathbf{X}^*$ of the DACMEE algorithm. The fact that the columns of $\mathbf{X}^*$ can only be eigenvectors of $\mathbf{R}_{yy}$ will then follow from this necessary condition.

In the second part, we will prove that $\hat{\mathbf{X}}$ is indeed an equilibrium point, based on the monotonic increase of the objective function $J(\mathbf{X})$ under the the DACMEE algorithm. In the last part, we will prove that $\hat{\mathbf{X}}$ is the only stable equilibrium point by using the necessary condition derived in part 1 and by pointing out that non-principal eigenvectors of $\mathbf{R}_{yy}$ do not satisfy the stability conditions.

Proof (part 1): If in iteration i of the DACMEE algorithm node q is the updating node, then $\overline{\mathbf{X}}_q^{i+1}$, as defined in Table I, contains the Q principal eigenvectors of $\overline{\mathbf{R}}_q^i = {\mathbf{V}_q^i}^H \mathbf{R}_{\tilde{y}_q \tilde{y}_q}^i \mathbf{V}_q^i$, and hence

$$\overline{\mathbf{R}}_q^i \overline{\mathbf{X}}_q^{i+1} = \overline{\mathbf{X}}_q^{i+1} \overline{\mathbf{\Sigma}}_q^i \tag{94}$$

where $\overline{\mathbf{\Sigma}}_q^i$ is a diagonal matrix containing the Q largest eigenvalues of $\overline{\mathbf{R}}_q^i$. By left-multiplying both sides with ${\mathbf{\Lambda}_q^i}^H$, plugging in (20) and (34), and using (29), we obtain

$$ {\mathbf{C}_q^i}^H \mathbf{R}_{yy} \mathbf{C}_q^i \mathbf{V}_q^i \overline{\mathbf{X}}_q^{i+1} = {\mathbf{\Lambda}_q^i}^H \overline{\mathbf{X}}_q^{i+1} \overline{\mathbf{\Sigma}}_q^i \, . \tag{95} $$

We define $\widetilde{\mathbf{X}}_q^{i+1}$ as

$$ \widetilde{\mathbf{X}}_q^{i+1} = \mathbf{V}_q^i \overline{\mathbf{X}}_q^{i+1} \tag{96} $$

such that (95) can be written as (again using (29))

$$ {\mathbf{C}_q^i}^H \mathbf{R}_{yy} \mathbf{C}_q^i \widetilde{\mathbf{X}}_q^{i+1} = {\mathbf{\Lambda}_q^i}^H \mathbf{\Lambda}_q^i \widetilde{\mathbf{X}}_q^{i+1} \overline{\mathbf{\Sigma}}_q^i \, . \tag{97} $$

Note that part of the variables in this equation are defined by $\mathbf{X}^i$, and part of them are defined by $\mathbf{X}^{i+1}$.

We now assume that $\mathbf{X}^i \in \mathcal{X}^*$, i.e., $\mathbf{X}^i$ is an equilibrium point. This means that $\mathbf{X}^{i+1} = \mathbf{X}^i$, and from (96), (35) and (38) it follows that $\widetilde{\mathbf{X}}_q^{i+1} = \left[{\mathbf{X}_q^i}^T \, \mathbf{I}_Q \, \ldots \, \mathbf{I}_Q \right]^T$. Therefore, in equilibrium we have that

$$ {\mathbf{C}_q^i}^H \mathbf{R}_{yy} \mathbf{X}^i = {\mathbf{\Lambda}_q^i}^H \mathbf{\Lambda}_q^i \begin{bmatrix} \mathbf{X}_q^i \\ \mathbf{I}_Q \\ \vdots \\ \mathbf{I}_Q \end{bmatrix} \overline{\mathbf{\Sigma}}_q^i \, . \tag{98} $$

Selecting the first M_q rows in (98), and relying on (14)-(16) and the fact that the first $M_q \times M_q$ diagonal block of $\mathbf{\Lambda}_q^i$ is equal to $\mathbf{I}_{M_q}$ (see (28)), yields

$$ \left[\mathbf{O}_{M_q \times \sum_{j=1}^{k-1} M_j} \, \mathbf{I}_{M_q} \, \mathbf{O}_{M_q \times \sum_{j=k+1}^{K} M_j} \right] \mathbf{R}_{yy} \mathbf{X}^i = \mathbf{X}_q^i \overline{\mathbf{\Sigma}}_q^i \, . \tag{99} $$

Furthermore, after left-multiplying (98) with the matrix $\left[{\mathbf{X}_q^i}^H \, \mathbf{I}_Q \, \ldots \, \mathbf{I}_Q \right]$, and using (26)-(28) and the fact that $\sum_{k \in \mathcal{K}} \mathbf{D}_k^i = {\mathbf{X}^i}^H \mathbf{X}^i = \mathbf{I}_Q$, we obtain

$$ {\mathbf{X}^i}^H \mathbf{R}_{yy} \mathbf{X}^i = \overline{\mathbf{\Sigma}}_q^i \, . \tag{100} $$

If an equilibrium is reached, then $\mathbf{X}^{i+1} = \mathbf{X}^i$ for all updating nodes $q \in \mathcal{K}$, hence the same reasoning can be performed for all $q \in \mathcal{K}$. By stacking the K matrix equations as defined in (99), $\forall \, q \in \mathcal{K}$, we have

$$ \mathbf{R}_{yy} \mathbf{X}^i = \begin{bmatrix} \mathbf{X}_1^i \overline{\mathbf{\Sigma}}_1^i \\ \vdots \\ \mathbf{X}_K^i \overline{\mathbf{\Sigma}}_K^i \end{bmatrix} \, . \tag{101} $$

Furthermore, since the lefthand side of (100) is independent of q, this shows that $\overline{\mathbf{\Sigma}}_q^i = \overline{\mathbf{\Sigma}}_k^i = \overline{\mathbf{\Sigma}}^i$, $\forall \, k, q \in \mathcal{K}$. Therefore, (101) is equal to

$$ \mathbf{R}_{yy} \mathbf{X}^i = \mathbf{X}^i \overline{\mathbf{\Sigma}}^i \, . \tag{102} $$

It is noted that ${\mathbf{X}^i}^H \mathbf{X}^i = \mathbf{I}_Q$, $\forall \, i \in \mathbb{N}$ and $\overline{\mathbf{\Sigma}}^i$ is a diagonal matrix. Therefore, and since we have assumed that $\mathbf{X}^i$ is an arbitrary equilibrium point in $\mathcal{X}^*$, we conclude that any equilibrium point can only contain eigenvectors of $\mathbf{R}_{yy}$. Note that this is a necessary condition for $\mathbf{X}^i$ to be an equilibrium point, but not a sufficient condition.

Proof (part 2): The fact that $\hat{\mathbf{X}} \in \mathcal{X}^*$ follows straightforwardly from the fact that $\hat{\mathbf{X}}$ maximizes (6)-(7), and the fact that the DACMEE algorithm results in a monotonic increase of $J(\mathbf{X})$ under the constraint (7). Note that the assumption (40) also assures that (6)-(7) has a unique maximum. Even if this assumption does not hold, i.e. if $\hat{\mathbf{X}}$ is not unique, the fix in Appendix VII-D will ensure that $\hat{\mathbf{X}} \in \mathcal{X}^*$, i.e., $\hat{\mathbf{X}}$ does not change under the DACMEE updates.

Proof (part 3): Finally, we prove that $\hat{\mathbf{X}}$ is the only stable equilibrium point. An equilibrium point $\mathbf{X}^*$ is stable under the DACMEE update rules if and only if any infinitesimal rotation of $\mathbf{X}^*$ does not increase the objective function $J(\mathbf{X})$, i.e.,

$$ \exists \delta > 0, \forall \mathbf{Q} \in \mathbb{U}^{M \times M} : \| \mathbf{Q} - \mathbf{I}_M \|_F \leq \delta \Rightarrow J(\mathbf{Q}\mathbf{X}^*) \leq J(\mathbf{X}^*) \tag{103} $$

where $\mathbb{U}^{M \times M}$ is the set of $M \times M$ unitary matrices. Indeed, since J is monotonically increasing under the DACMEE algorithm (Theorem III.1), equilibrium points $\mathbf{X}^*$ that do not satisfy (103) are unstable under the DACMEE update rules since a small perturbation may cause $J(\mathbf{Q}\mathbf{X}^*) \geq J(\mathbf{X}^*)$, and since $J\left(\mathbf{X}^{i+1}\right) \geq J\left(\mathbf{X}^i\right)$ for all subsequent iterations $i \in \mathbb{N}$, the DACMEE algorithm cannot return to the original equilibrium point $\mathbf{X}^*$. If $\mathbf{X}^*$ contains eigenvectors that are not in $\hat{\mathbf{X}}$, it does not satisfy (103). Therefore, $\hat{\mathbf{X}}$ is the only point that both satisfies the necessary condition for equilibria (102) and the stability condition (103), and hence it is the only stable equilibrium point in $\mathcal{X}^*$.

C. Proof of Theorem III.4

Proof outline: The proof relies on the necessary condition for an equilibrium point, as derived in the proof of Theorem III.3. The main idea is to add an error term in the SOE that defines this necessary condition, to indicate that the equilibrium point has not been reached yet. We will then use the result of Theorem III.2 to show that, for $i \to \infty$, this error term vanishes. The resulting limit-equation will show that the columns of $\mathbf{X}^i$ should converge to eigenvectors of $\mathbf{R}_{yy}$.

Proof: The proof of Theorem III.3 relies on the fact that $\mathbf{X}^i \in \mathcal{X}^*$, which was used to obtain (98) from (97). However, if $\mathbf{X}^i \notin \mathcal{X}^*$, then $\mathbf{X}^{i+1} \neq \mathbf{X}^i$ and therefore an error term $\mathbf{E}^i$ should be added in (98), i.e.,

$$ {\mathbf{C}_q^i}^H \mathbf{R}_{yy} \mathbf{X}^i = {\mathbf{\Lambda}_q^i}^H \mathbf{\Lambda}_q^i \begin{bmatrix} \mathbf{X}_q^i \\ \mathbf{I}_Q \\ \vdots \\ \mathbf{I}_Q \end{bmatrix} \overline{\mathbf{\Sigma}}_q^i + \mathbf{E}^i \, . \tag{104} $$

Therefore, an error term also appears in (100) and (101), which are derived from (98), i.e.,

$$ \mathbf{R}_{yy} \mathbf{X}^i + \mathbf{\Delta}^i = \begin{bmatrix} \mathbf{X}_1^i \overline{\mathbf{\Sigma}}_1^i \\ \vdots \\ \mathbf{X}_K^i \overline{\mathbf{\Sigma}}_K^i \end{bmatrix} \tag{105} $$

and

$$ {\mathbf{X}^i}^H \mathbf{R}_{yy} \mathbf{X}^i + \mathbf{\Delta}_q^i = \overline{\mathbf{\Sigma}}_q^i \tag{106} $$

where $\mathbf{\Delta}^i$ and $\mathbf{\Delta}_q^i$, $\forall \, q \in \mathcal{K}$, are error terms. However, from Theorem III.2, it follows that the error $\mathbf{E}^i$ vanishes in (104)

if $i \to \infty$, and therefore

$$\lim_{i\to\infty} \|\mathbf{X}^{i\,H}\mathbf{R}_{yy}\mathbf{X}^i - \overline{\mathbf{\Sigma}}_k^i\| = 0\,,\ \forall\, k \in \mathcal{K} \tag{107}$$

$$\lim_{i\to\infty} \|\mathbf{R}_{yy}\mathbf{X}^i - \mathbf{X}^i\overline{\mathbf{\Sigma}}^i\| = 0 \tag{108}$$

where $\overline{\mathbf{\Sigma}}^i = \mathbf{X}^{i\,H}\mathbf{R}_{yy}\mathbf{X}^i$. Since $\overline{\mathbf{\Sigma}}_k^i$, $\forall\, k \in \mathcal{K}$, are diagonal matrices (by construction), it follows from (107) that $\lim_{i\to\infty}\overline{\mathbf{\Sigma}}^i$ is also diagonal. This fact, together with (108) shows that the columns of $\mathbf{X}^i$ converge to eigenvectors of $\mathbf{R}_{yy}$ when $i \to \infty$. Furthermore, it follows from Theorem III.2 that $\mathbf{X}^i$ cannot jump between different eigenvectors over the different iterations if $i \to \infty$, which proves the theorem.

D. Algorithm fixes for special cases

1) Rank deficient $\mathbf{\Lambda}_q^i$: In the rare case where $\mathbf{\Lambda}_q^i$ is rank deficient, $\mathbf{V}_q^i$ does not exist such that (34) cannot be computed. Rank deficiency of $\mathbf{\Lambda}_q^i$ occurs when there is a node k for which $\mathbf{X}_k^i$ has linearly dependent columns (assuming $M_k > Q$). There are two ways to circumvent this problem:

- The linearly dependent columns in $\mathbf{X}_k^i$, and the corresponding entries in $\mathbf{z}_k^i$ are removed when constructing $\mathbf{R}_{\tilde{y}_q\tilde{y}_q}^i$ at the updating node q. As the removed entries can be reconstructed from the other entries in $\mathbf{z}_k^i$, this removal cannot counteract the monotonic increase of $J(\mathbf{X}_k^i)$.
- Node k replaces the linearly dependent columns in $\mathbf{X}_k^i$ with random columns, yielding a new $\mathbf{X}_k^i$ in which all the columns are linearly independent. Note that this replacement provides the other nodes with additional information, and therefore this is a better option than the previous one.

2) Degenerated principal eigenvalues: In the rare case where the n-th largest eigenvalue of $\overline{\mathbf{R}}_q^i$ is degenerate, i.e. $\overline{\lambda}_n^i = \overline{\lambda}_{n+1}^i$ (with $n \leq Q$), $\overline{\mathbf{X}}_q^{i+1}$ is ill-defined in its n-th and $(n+1)$-th column. For the sake of an easy exposition and w.l.o.g., we consider the worst case, i.e., we assume that all columns of $\overline{\mathbf{X}}_q^{i+1}$ are ill-defined, i.e., $\overline{\lambda}_1^i = \overline{\lambda}_2^i = \ldots = \overline{\lambda}_{Q+1}^i$. In this case, the eigenvectors corresponding to these eigenvalues span a subspace $\mathcal{S}$ with a dimension of $(Q+1)$. A unique $\overline{\mathbf{X}}_q^{i+1}$ can then be defined as

$$\overline{\mathbf{X}}_q^{i+1} = \arg\min_{\overline{\mathbf{X}}} \|\overline{\mathbf{X}} - \mathbf{P}_q^i\|_F \tag{109}$$

$$\text{s.t. } \cdot\ \text{Range}\{\overline{\mathbf{X}}\} \in \mathcal{S} \tag{110}$$

$$\cdot\ \overline{\mathbf{X}}^H\overline{\mathbf{X}} = \mathbf{I}_Q \tag{111}$$

where $\mathbf{P}_q^i$ is defined in (68). Note that this also ensures convergence, since $\overline{\mathbf{X}}_q^{i+1} = \mathbf{P}_q^i$ implies that $\mathbf{X}^{i+1} = \mathbf{X}^i$ (this can be seen from (35) and (38)). It is noted that (109) also resolves the scaling ambiguity in the eigenvectors.

REFERENCES

[1] A. Bertrand and M. Moonen, "Distributed adaptive eigenvector estimation of the sensor signal covariance matrix in a fully connected sensor network," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)*, Vancouver, Canada, May 2013, pp. 4236–4240.

[2] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed., ser. Springer Series in Statistics. New York: Springer, 2002.

[3] Y.-A. Le Borgne, S. Raybaud, and G. Bontempi, "Distributed principal component analysis for wireless sensor networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008.

[4] M. Gastpar, P. Dragotti, and M. Vetterli, "The distributed Karhunen-Loève transform," in *IEEE Workshop on Multimedia Signal Processing*, dec. 2002, pp. 57 – 60.

[5] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, 1986.

[6] A. Scaglione, R. Pagliari, and H. Krim, "The decentralized estimation of the sample covariance," in *Asilomar Conference on Signals, Systems and Computers*, oct. 2008, pp. 1722 –1726.

[7] I. Markovsky and S. Van Huffel, "Overview of total least-squares methods," *Signal Processing*, vol. 87, no. 10, pp. 2283 – 2302, 2007, special Section: Total Least Squares and Errors-in-Variables Modeling.

[8] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[9] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4583–4588, 2009.

[10] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – part I: sequential node updating," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5277–5291, 2010.

[11] ——, "Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2196–2210, May 2011.

[12] S. Macua, P. Belanovic, and S. Zazo, "Consensus-based distributed principal component analysis in wireless sensor networks," in *International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, june 2010, pp. 1 –5.

[13] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Trans. Signal Processing*, vol. 59, no. 5, pp. 2320–2330, May 2011.

[14] ——, "Low-complexity distributed total least squares estimation in ad hoc sensor networks," *IEEE Transactions on Signal Processing*, vol. 60, pp. 4321–4333, Aug. 2012.

[15] Z.-J. Bai, R. H. Chan, and F. T. Luk, "Principal component analysis for distributed data sets with updating," in *Proceedings of the 6th international conference on Advanced Parallel Processing Technologies*, ser. APPT'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 471–483.

[16] L. Li, X. Li, A. Scaglione, and J. Manton, "Decentralized subspace tracking via gossiping," in *Distributed Computing in Sensor Systems*, ser. Lecture Notes in Computer Science, R. Rajaraman, T. Moscibroda, A. Dunkels, and A. Scaglione, Eds. Springer Berlin Heidelberg, 2010, vol. 6131, pp. 130–143.

[17] L. Li, A. Scaglione, and J. Manton, "Distributed principal subspace estimation in wireless sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 725–738, 2011.

[18] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore: The Johns Hopkins University Press, 1996.

[19] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, vol. 15, no. 3, pp. 267–273, 1982.